Proyecto de fin de Carrera
Ingeniería en Informática
Curso 2014/2015

Diseño, análisis e implementación de una plataforma Web para la gestión y administración de centralitas VoIP multiempresa

Luis Miguel Alonso Gan

Director: Roberto Casas Millán

Ponente: José Luis Briz Velasco

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Junio de 2015

Diseño, análisis e implementación de una plataforma Web para la gestión y administración de centralitas VoIP multiempresa

RESUMEN

Las centralitas de teléfono son cada vez más importantes tanto en empresas grandes como pequeñas. Poder estar disponible y localizado en horario de trabajo, dividir en grupos de personas, no perder ningún cliente, informar sobre horarios, etc. las convierten en algo necesario.

Entre las ventajas de la VoIP destacan la flexibilidad de disponer de un número de teléfono en cualquier parte sin una línea física, y el precio, con tarifas más baratas que las líneas ofrecidas por compañías telefónicas.

Debido a los elevados precios de las centralitas telefónicas que se ofertan, normalmente las pequeñas y medianas empresas no pueden permitirse adquirir estos servicios. Aquí es donde entra en juego Asterisk.

Asterisk es un programa de software libre (bajo licencia GPL) que proporciona funcionalidades de una central telefónica (PBX). Como cualquier PBX, se puede conectar un número determinado de teléfonos para hacer llamadas entre sí e incluso conectar a un proveedor de VoIP o bien a una RDSI tanto básicos como primarios.

En este proyecto se ha desarrollado un frontal Web desde donde se puede configurar y gestionar una centralita Asterisk de una manera sencilla e intuitiva. Se ha pensado en una aplicación útil tanto para administradores como para los usuarios que quieran gestionarse su cuenta.

El objetivo es ofrecer un servicio de centralita virtual multiempresa que sea modular, el cliente sólo tiene aquellos servicios que necesita y por tanto paga sólo por los servicios prestados, y escalable, puede aumentar el número de líneas sin tener que instalar nuevas líneas RDSI (evitando la instalación hardware más el contrato de líneas con la compañía telefónica).

Para el desarrollo de todas estas tareas se ha utilizado como lenguaje de programación PHP, combinado con javaScript/AJAX, así como sistemas de carga asíncrona de contenidos usando jQuery. Para el diseño Web HTML5 y CSS3.

Para la gestión del proyecto se ha usado metodología Scrum en una herramienta interna de la empresa y como gestor de versiones Subversión.

Agradecimientos

A mis padres, a mi hermana Noelia, a Isabel, a familiares y amigos por apoyarme todos estos años y acompañarme en este recorrido.
A mis jefes Luis y Roberto por enseñarme y guiarme durante el desarrollo del proyecto.
A mi ponente José Luis Briz por su paciencia, disposición y ayuda durante el desarrollo de esta memoria.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El presente documento representa la memoria de trabajo del proyecto de final de carrera "Diseño, análisis e implementación de una plataforma Web para la gestión y administración de centralitas VoIP multiempresa", realizado por el alumno Luis Miguel Alonso Gan, cuyo ponente ha sido José Luis Briz Velasco y su director Roberto Casas Millán. Se ha desarrollado dentro de la empresa Diaple Networking S.L.

1.1. Objetivos y alcance

En este proyecto se ha desarrollado una plataforma accesible a través de la web desde donde se pueda configurar y gestionar la centralita de una manera sencilla e intuitiva tanto para el usuario como para los administradores. La idea es crear un servicio de centralita virtual que albergue distintas empresas y mediante un contrato, ofrecerles sólo aquellos servicios que necesiten. Esto permite al cliente crecer de manera escalable sin tener que hacer un gran desembolso inicial. En caso de necesitar aumentar el número de líneas esto evita instalar más componentes hardware (tarjetas *RDSI*) o aumentar el contrato con la operadora de las líneas *RDSI* correspondientes, resultando por tanto una alternativa más económica.

Además a través de la base de datos, se pueden parametrizar nuevas aplicaciones (funcionalidades) sin tener que modificar los ficheros de la centralita. Es decir, mediante el uso de la base de datos es posible añadir este tipo de aplicaciones que se encargarán de interactuar con Asterisk para realizar las instrucciones programadas dejándolas integradas y accesibles desde la aplicación Web (por ejemplo: saber el tiempo de duración de una llamada, colgar una llamada, lanzar una llamada de forma automática, transferir una llamada, etc.).

El fin es que desde cualquier tipo de dispositivo y lugar (siempre que tenga conexión a internet) se pueda realizar una consulta o un cambio sin tener que

estar en la oficina.

1.2. Motivación

Las centralitas de teléfono son cada vez más importantes tanto en empresas grandes como pequeñas. Poder estar disponible y localizado en horario de trabajo, dividir en grupos de personas, no perder ningún cliente, informar sobre horarios, etc. las convierten en algo necesario.

Entre las distintas ventajas de la *VoIP* destacan la flexibilidad de poder disponer de un número de teléfono en cualquier parte del mundo sin una línea física, y el precio, con unas tarifas mucho más baratas que las líneas tradicionales ligadas a compañías telefónicas.

Debido a los elevados precios de las centralitas telefónicas que se ofertan, normalmente las pequeñas y medianas empresas no pueden permitirse adquirir estos servicios. Aquí es donde entra en juego Asterisk.

Asterisk es un programa de software libre (bajo licencia *GPL*) que proporciona funcionalidades de una central telefónica (*PBX*). Como cualquier *PBX*, se puede conectar un número determinado de teléfonos para hacer llamadas entre sí e incluso conectar a un proveedor de *VoIP* o bien a una *RDSI* tanto básica como primaria.

1.3. Metodología de desarrollo

Para el desarrollo de este proyecto se ha utilizado metodología *Scrum*. La decisión de elegir esta metodología se ha debido a que permite tener un buen control del producto en el tiempo. Facilita establecer un calendario de reuniones de seguimiento con el cliente, que hemos adoptado como fechas de entrega de diferentes partes.

Scrum es una metodología ágil para la gestión de proyectos. Su principal objetivo es obtener resultados (normalmente prototipos) cuanto antes y adaptarse a los cambios (normalmente, los cambios en los requisitos).

En *Scrum* un proyecto se ejecuta en bloques temporales cortos y fijos llamados *sprint*. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

En nuestro caso los *sprint* fueron definidos cada dos semanas. Las reuniones con el cliente se realizaban el primer día y posteriormente, se definían las tareas y objetivos para el próximo *sprint*. Asimismo, cada día se realizaban reuniones de 5 minutos en el equipo de trabajo para comprobar que se había hecho y si se estaban cumpliendo los tiempos de plazo.

1.4. Estructura de la memoria

El primer capítulo describe los objetivos y alcance del proyecto para introducir al lector en la memoria. El segundo capítulo define los conceptos subyacentes que se van a nombrar a lo largo de la memoria y que es interesante que el lector conozca qué son. El tercer capítulo explica de manera resumida el trabajo realizado en la elaboración del proyecto. El cuarto capítulo presenta las conclusiones del proyecto y las posibles líneas futuras a partir del presente PFC. Los anexos A y B, amplían información referente a la planificación, el análisis y diseño en detalle del sistema, y los anexos C y D, recogen los manuales de usuario y técnico respectivamente.

1.5. Marco temporal del proyecto

En el diagrama de la Fig. 1.1 se resume el desarrollo temporal del proyecto. También se incluye en la tabla 1.1 un resumen de las horas que han sido necesarias para completar cada una de las distintas fases del desarrollo del proyecto.

El control y organización del desarrollo del proyecto está explicado más en profundidad en el anexo A -Planificación y control de esfuerzos-.

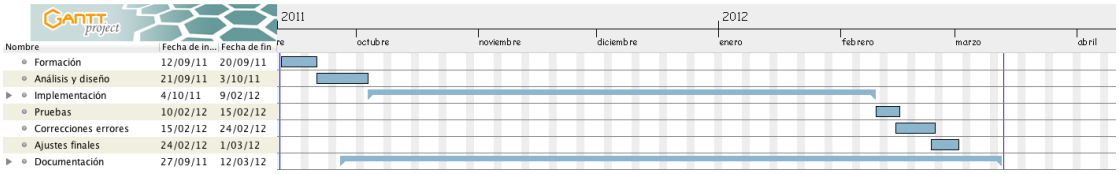


Figura 1.1: Diagrama de la distribución temporal de las fases del proyecto

Tarea	Horas
Formación	35
Reuniones	10
Análisis y diseño	35
Implementación	315
Pruebas	20
Corrección de errores	15
Ajustes finales	10
Documentación	110

Tabla 1.1: Horas empleadas en cada tarea

Capítulo 2

Conceptos subyacentes

2.1. Centralita IP

Una central telefónica *IP* es un equipo telefónico diseñado para ofrecer servicios de comunicación a través de las redes de datos. A esta aplicación se le conoce como voz por *IP (VoIP)*, donde la dirección *IP (Internet Protocol)* es la identificación de los dispositivo dentro de la Web. Con los componentes adecuados se puede controlar un número ilimitado de anexos en sitio o remotos vía internet, añadir vídeo, conectarle troncales digitales o servicios de *VoIP (SIP trunking)* para llamadas internacionales a bajo coste. Los aparatos telefónicos que se usan se les llama teléfonos *IP* o *SIP* y se conectan a la red. Además por medio de puertos de enlaces se le conectan las líneas normales de las redes telefónicas públicas, y anexos analógicos para teléfonos estándar (fax, inalámbricos, contestadores, etc.).

Para las empresas internacionales estas centrales se han convertido en un equipo indispensable. La existencia de sistemas operativos y software gratuito han hecho aumentar mucho la instalación de centrales IP, algunos incluso usan un PC como hardware. La seguridad es algo muy importante ya que al estar conectados a Internet existe el riesgo de ser atacados por hackers, virus, etc.

Las aplicaciones de esta tecnología están en continuo desarrollo y hacen que sea sencillo crear y desplegar una amplia gama de aplicaciones de telefonía y servicios, incluyendo los de una *PBX* con diversas pasarelas (gateways) de *VoIP*. Se han liberado los códigos bajo la licencia *GNU General Public License (GPL)*, y están disponibles para su descarga en forma gratuita.

Claramente este es el futuro, cada vez son más las pequeñas y medianas empresas que cuentan con esta tecnología con total confianza y una buena calidad de audio.

2.2. Asterisk

Asterisk [1, 2] es un programa de software libre (bajo licencia *GPL*) que proporciona funcionalidades de una central telefónica (*PBX*). Como cualquier *PBX*, se puede conectar un número determinado de teléfonos para hacer llamadas entre sí e incluso conectar a un proveedor de *VoIP* o bien a una *RDSI* tanto básicos como primarios.

El programa de software Asterisk fue desarrollado por Mark Spencer, por entonces estudiante de ingeniería informática en la Universidad de Auburn, Alabama. Mark había creado en 1999 la empresa "Linux Support Services" con el objetivo de dar soporte a usuarios de *GNU/Linux*. Para ello necesitaba una central telefónica, pero ante la imposibilidad de adquirirla dados sus elevados precios, decidió construir una con un PC bajo Linux, utilizando lenguaje C.

Posteriormente "Linux Support Services" se convertiría en el año 2001 en "Digium", redirigiendo sus objetivos al desarrollo y soporte de Asterisk.

Asterisk actualmente también se distribuye en versiones para los sistemas operativos *BSD, Mac OS X, Solaris* y *Microsoft Windows*, aunque la plataforma nativa (*GNU/Linux*) es la que cuenta con mejor soporte de todas.

Asterisk incluye muchas características que anteriormente sólo estaban disponibles en costosos sistemas propietarios *PBX*, como buzón de voz, conferencias, *IVR*, distribución automática de llamadas, y otras muchas. Los usuarios pueden crear nuevas funcionalidades escribiendo un dialplan en el lenguaje de script de Asterisk o añadiendo módulos escritos en lenguaje C o en cualquier otro lenguaje de programación soportado en *GNU/Linux*.

Para conectar teléfonos estándares analógicos son necesarias tarjetas electrónicas telefónicas *FXS o FXO* fabricadas por Digium u otros proveedores, ya que para conectar el servidor a una línea externa no basta con un simple módem.

Quizá lo más interesante de Asterisk es que reconoce muchos protocolos *VoIP* como pueden ser *SIP, H.323, IAX y MGCP*. Asterisk puede interoperar con terminales *IP* actuando como un registrador y como *gateway* entre ambos.

Uno de los puntos fuertes del software Asterisk es que permite la unificación de tecnologías: *VoIP, GSM y PSTN*.

2.3. Model-View-Controller (MVC)

MVC es un patrón o modelo de abstracción de desarrollo del software que separa los datos de una aplicación, interfaz de usuario y lógica de negocio. Fue descrito por primera vez en 1979 por *Trygve Reenskaugm* [3], por aquel entonces trabajador en Smalltalk en los laboratorios de Xerox. Cuando hablamos de patrón nos referimos a una manera de estructurar los códigos de la aplicación que permiten

mejorar la modularización de un sistema y el mantenimiento del mismo.

Un Model-View-Controller debe proporcionar métodos de seguridad en acceso a base de datos, separación de las vistas respecto a la lógica de negocio, así como también la recepción de eventos de los clientes que interactúan con la interfaz de usuario.

La **vista** es todo lo referente a la interfaz gráfica de una aplicación, en Internet, vendrían a ser los contenidos html renderizados por un navegador web.

El **controlador** es el módulo, dentro de un patrón *MVC* que se encarga de capturar todas las peticiones realizadas por los usuarios de un software, en este caso, a través del navegador web. Toda petición es capturada por el controlador que tiene la posibilidad de interactuar con el Modelo de Datos para extraer información eventualmente y mandarlo a las vistas, que son finalmente renderizadas en un navegador. Eventualmente se requiere información de la Base de Datos pero no se desea realizar costosas peticiones http que ralentizan el funcionamiento de una aplicación web. Se dispone de la Tecnología *Asynchronous JavaScript And XML* (AJAX), en la que mediante peticiones al controlador, devuelve resultados en diferentes formatos que son tratados en segundo plano y mostrados en la interfaz, aumentando el rendimiento de la aplicación y permitiendo una interacción con el cliente más cercana.

El **modelo** es el encargado de tener la estructura de datos de cada una de las tablas de la Base de Datos. En primera instancia, tiene los métodos de consulta, actualización de campos, y métodos propios del modelo. Toda esta jerarquía en la estructura de la aplicación ayuda al mantenimiento y modularidad de la aplicación, y hoy en día es uno de los patrones de diseño de aplicaciones web más usado. La Fig. 2.1 muestra el patrón genérico de *MVC* con todas las posibles combinaciones de comunicación entre módulos.

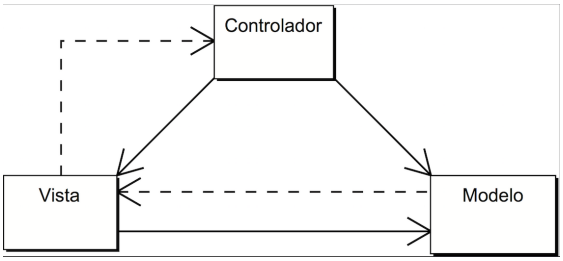


Figura 2.1: Modelo - Vista - Controlador

Capítulo 3

Descripción

A lo largo de este capítulo se explicará el trabajo desempeñado y como está estructurada la aplicación a nivel de código.

El sistema desarrollado se trata de un frontal Web para configurar, gestionar y operar con un entorno Asterisk. Su principal función es facilitar a los administradores la configuración de la centralita (*PBX*) de una empresa, y a los usuarios la consulta de agenda, su registro de llamadas, buzón de voz o el cambio de comportamiento de sus dispositivos (por ejemplo, el teléfono de la mesa de su puesto de trabajo). La gran diferencia entre esta implantación y la mayoría de las que se ofrecen en el mercado es que los dispositivos están asociados a cuentas, no a extensiones. De este modo un usuario puede tener varias extensiones, y con una sola extensión puede tener más de una forma de contacto.

En la Fig. 3.1 se resume qué componentes del proyecto he realizado íntegramente (en color verde oscuro), cuáles en colaboración (verde claro), y cuáles se han construido integrando otros ya existentes (en blanco).

El orden de realización fue prácticamente como se muestra de izquierda a derecha en la figura. El proyecto se estructuró en tres fases principales, separadas por reuniones de decisión y seguimiento, en las que se definía cada una de las siguientes fases (análisis, base de datos e interfaz):

- Usuarios, roles y permisos.
- Agenda, registro de llamadas y buzón de voz.
- Configuración de la centralita.

De esta manera, cada una o dos semanas se entregaban partes de la aplicación demostrables al cliente, y por otro lado nuevas funcionalidades para testar dentro del equipo de trabajo.

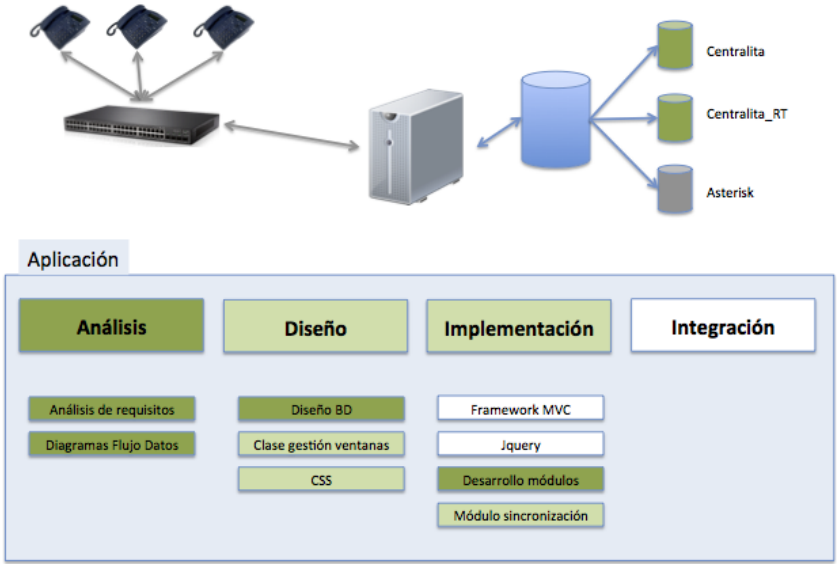


Figura 3.1: Gráfico trabajo realizado

3.1. Diseño

El diseño de esta aplicación surge de las nuevas vías abiertas por *HTML5* [4] y *CSS3* [5], que junto con *JavaScript* y *JQuery* [6, 7] actualizado permiten mover elementos dentro de una página Web (*JQuery UI Draggable plugin*).

El diseño elegido se inspira en los escritorios de los sistemas operativos más extendidos (Windows, Linux) con su barra de herramientas, sus iconos, sus ventanas y sus Widgets. Este diseño aporta una visualización agradable, es fácil de usar, tiene una curva de aprendizaje rápida dada la familiaridad de los usuarios con estos interfaces, y además presenta mejoras frente a otros diseños como, por ejemplo, la visualización de parámetros de configuración con ventanas abiertas de manera simultánea.

La Fig. 3.2 muestra el aspecto general una vez se ha accedido a la aplicación mediante usuario y contraseña.

En la parte superior izquierda de la figura pueden verse los accesos directos a cada apartado de la aplicación, que serán visibles o no dependiendo de los permisos asociados al rol del usuario que ha accedido al sistema. El ejemplo que se ha puesto en la figura tiene un rol de administrador, que da acceso a todos los apartados. El rol de super-administrador da acceso a todas las empresas simultáneamente.

A la derecha se muestra un Widget con distintos apartados que se actualizan en tiempo real. Por un lado se encuentra las formas de contacto que posee el usuario, es decir, los medios a través de los cuales le van a llegar las llamadas al usuario

Figura 3.2: Diseño escritorio

(teléfono fijo, softphone en el ordenador, móvil etc.). Por otro lado tenemos las colas a las que pertenece el usuario. Normalmente a cada departamento se le asigna una cola, pero la aplicación permite que un usuario ligado a varios departamentos figure en diferentes colas. Por último, tenemos las llamadas en cola que está gestionando el usuario. Si en ese momento se encuentra hablando con un cliente mostrará los segundos que lleva hablando, y si tiene una llamada en espera, saldrá con el estado en espera.

En la parte inferior se sitúa la barra de estado. En primer lugar aparece el logotipo de la empresa Diaple Networking S.L., como imagen de marca de la empresa desarrolladora. A continuación, se acumulan las diferentes ventanas que tenemos en uso en ese momento (minimizadas o no), pudiendo interactuar con todas ellas en cualquier momento desde la barra de estado hasta que se cierran completamente. En la derecha de la barra de estado, se muestra el icono de una terminal, sólo para los usuarios administradores, que activa una ventana con un terminal Linux embebido conectado a la centralita. Permite gestionar y analizar el comportamiento de la centralita por medio de órdenes desde la misma aplicación. Detrás de este icono, o en caso de no ser administrador en primer lugar, aparece la empresa y a continuación el usuario con que se ha entrado a la aplicación.

Si es un super-administrador el que accede a la aplicación, el escritorio muestra inicialmente sólo dos iconos — Fig. 3.3—. El primero `Configuración global` permite administrar las empresas y los números de teléfonos contratados con el proveedor de *VoIp* (denominados *DIDs* en la aplicación), con la posibilidad de asociar un *DID* a una empresa. El segundo icono `Salir` finaliza la aplicación.

Figura 3.3: Diseño escritorio super-administrador

La parte inferior derecha de la barra de herramientas, que antes mostraba la compañía a la que pertenecía el usuario, ahora ofrece un desplegable para poder seleccionar y configurar cualquiera de las empresas dadas de alta en la aplicación sin tener que terminar e iniciar sesión. Esta característica tiene mucha importancia debido a que la mayoría de las empresas carecen de personal en plantilla para gestionar su centralita, y junto con el producto de centralita, contratan el servicio de mantenimiento para que les configuremos los cambios que necesiten desde nuestras instalaciones.

3.1.1. Gestión de las ventanas

Para poder gestionar las ventanas, se creó una clase en código *JavaScript* `dialogManager` que no sólo aprovecha algunas propiedades de los diálogos de *JQuery* sino que añade funcionalidades nuevas. La más característica es que para poder minimizar y volver a restaurar la ventana, así como para poder evitar que se abra una nueva que ya teníamos abierta pero que no veíamos, necesitamos que tenga un identificador único. De esta manera cuando se crea la ventana se le asigna un identificador único, se añade a una pila y se puede acceder desde la barra inferior del escritorio, hasta que se cierra -y se borra de la pila- pulsando la "X". Se ha conseguido así un interfaz en el que el usuario opera con las ventanas tal como lo hace en los entornos de sobremesa habituales.

A continuación se muestra una captura con varias ventanas abiertas para que se pueda apreciar su manejo simultáneo y el aspecto que toma la barra inferior.

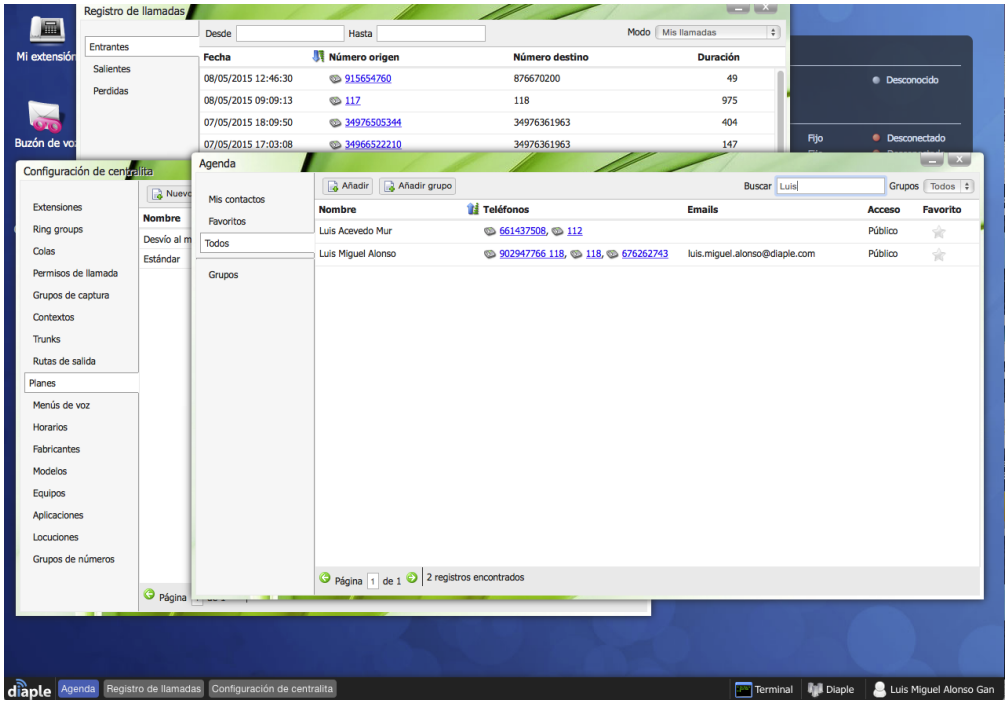


Figura 3.4: Diseño escritorio con ventanas

Las ventanas se agrupan en la barra inferior siguiendo un orden de izquierda a derecha en orden de apertura. Se ilumina en azul la que está utilizándose mientras que el resto se mantiene en gris. El cambio de ventana activa se efectúa pulsando sobre la que se desea o sobre su correspondiente recuadro en la barra inferior.

Aprovechando las cualidades de *HTML5*, y para hacer más rápida y cómoda las labores de los administradores, se introdujo en algunas ventanas la opción de arrastrar y soltar para cambiar el orden en que se ejecutan las acciones.

Otra característica disponible para los usuarios es poder escuchar los buzones de voz desde la aplicación. En algunos navegadores al principio del proyecto no estaban disponibles todas las funciones de *HTML5*, como la reproducción de audio o vídeo en formato *HTML5*, es decir sin uso de Flash, por lo que comprobamos primero el navegador desde el que se está ejecutando. Según soporte o no la reproducción de audio, ejecutamos el audio *HTML5* o cargamos el audio a través de Flash (que requiere *Adobe Flash Player*). De este modo no se obliga al cliente a que se instale una versión concreta de un navegador Web para poder escuchar la reproducción de los mensajes que les han dejado en el buzón de voz. Actualmente todos los navegadores Web conocidos disponen de esta opción, pero a fin de ofrecer la máxima portabilidad, la aplicación se puede configurar para que la

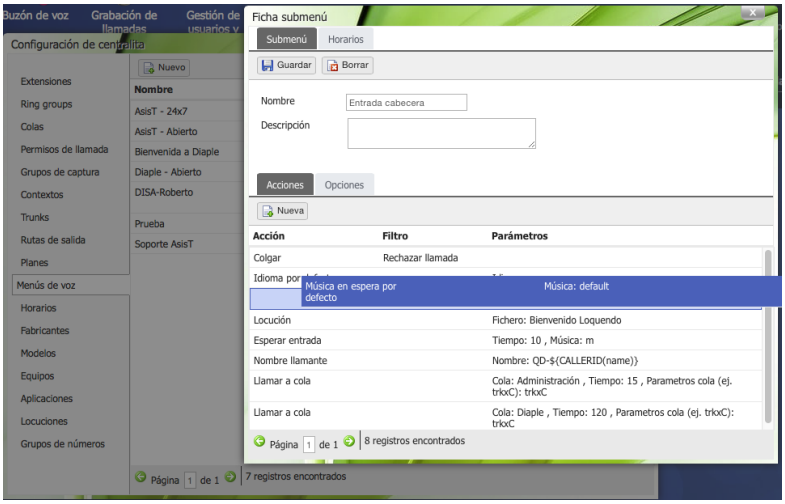


Figura 3.5: Diseño arrastrar y soltar

centralita mande los mensajes del buzón de voz por correo, reproducibles desde nuestro cliente de correo electrónico, desde el PC o desde el móvil.

3.2. Patrón

El sistema ha sido desarrollado mediante el uso del patrón de diseño Modelo Vista Controlador (*MVC*) con el uso de un *Framework* propio, que proporciona una estructura bien definida la cual facilita un buen desarrollo y organización del proyecto.

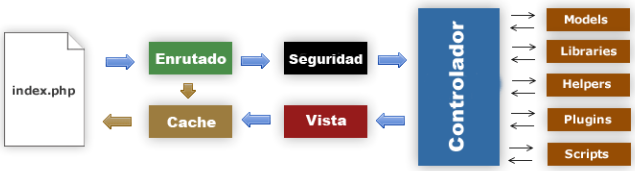


Figura 3.6: Arquitectura del Framework propietario

Un archivo de configuración define dónde se encuentran tanto los controladores como las vistas. Siguiendo el patrón de diseño *MVC*, existe una carpeta llamada *controller*, que contiene los archivos de gestión del controlador, y otra carpeta *views*, que incluye tantas subcarpetas como vistas existen (una por módulo).

En la figura 3.6 se muestra la arquitectura del patrón. Su descripción detallada se recoge en el apartado *Patrón MVC* del anexo D -Manual técnico-.

3.3. Estructura de la aplicación

La estructura detallada de la aplicación se explica en el anexo D, Manual técnico.

3.4. Base de datos

La base de datos de la aplicación se ha realizado en *MySQL* [8] por tres motivos. En primer lugar es muy rápida y se integra muy bien con *PHP* [9], en segundo es la que habitualmente utilizan en la empresa, y por último, la base de datos que se usa en nuestra instalación de Asterisk también utiliza *MySQL*. En definitiva nos facilita la gestión y sincronización de datos entre las BD de la aplicación y la de Asterisk.

El sistema está basado en tres bases de datos:

- **Base de datos de la aplicación:** Utilizada por el frontal Web.
- **Base de datos tiempo real:** Utilizada por la centralita: Asterisk consulta esta base de datos para ciertas operaciones.
- **Base de datos de Asterisk:** Utilizada por Asterisk.

Asterisk tiene su propia base de datos pero para ciertas funcionalidades necesita consultar la base de datos de la aplicación. Sin embargo, la base de datos de la aplicación está sujeta a modificaciones en cualquier momento, que no deben de afectar a Asterisk hasta que no se consoliden. Para desacoplar las operaciones de configuración del funcionamiento interno, hemos introducido una base de datos intermedia que denominamos *tiempo real*, evitando que Asterisk acceda directamente a la base de datos de la aplicación. Una vez finalizada la configuración, los cambios efectuados mediante el interfaz web en la base de datos de la aplicación se aplican a la de tiempo real sincronizándolos desde el menú `Aplicar Configuración`. La sincronización se realiza a nivel de empresa, es decir, no podemos configurar más de una empresa a la vez desde una misma interfaz Web. La Fig. 3.7 esquematiza este procedimiento.

La solución es intuitiva y parece simple pero esconde problemas complejos debidas sobre todo a dependencias entre módulos y a la comprobación de la integridad referencial. En los siguientes apartados explicamos cómo se ha solucionado este aspecto.

3.4.1. Subdivisión en módulos

Para permitir mayor control sobre los cambios, la configuración del sistema se subdivide en módulos. Los parámetros de cada módulo determinan qué se sincro-

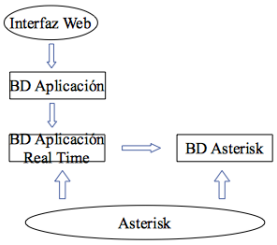


Figura 3.7: Flujo de datos sincronización

niza, qué acciones se ejecutarán tras la sincronización, y si es el caso si precisa la sincronización previa de otros módulos, y son los siguientes:

- **Tablas de la aplicación:** Cuando el módulo se sincroniza las tablas especificadas se sincronizan desde la base de datos de aplicación con la base de datos tiempo real.
- **Tablas de Asterisk:** *Queries* SQL para generar las tablas de Asterisk a partir de la consulta a la base de datos tiempo real. Además se pueden especificar las acciones a ejecutar tras la modificación de una tabla. La acción recibe como parámetros los cambios realizados en la tabla, de manera que es posible ejecutar acciones a nivel de registro modificado, permitiendo así diferenciar inserciones, modificaciones y borrados. El motivo de introducir acciones a nivel de registro modificado viene de la necesidad de ejecutar ciertas acciones en el AMI para algunas tablas.
- **Acción post-sincronización:** Acción ejecutada tras sincronizar las bases de datos. Por ejemplo generar ficheros, ejecutar comandos en el AMI de Asterisk etc.
- **Dependencias:** Módulos de los que depende un módulo dado. Una sincronización de un módulo implicará la sincronización de los módulos dependientes.

3.4.2. Funcionamiento interno

La clase abstracta `AsteriskModule` implementa el comportamiento común que tienen todos los módulos. Por un lado sincroniza tablas de aplicación y Asterisk, y por otro gestiona las dependencias entre módulos. El método que realiza la sincronización (`synchronize`), se encarga de estas funciones en siete fases:

1. Iniciar transacción en la base de datos.

2. Generar los datos de las tablas de Asterisk creadas por el modulo y sus dependencias en tablas temporales.

3. Sincronizar las tablas de aplicación del módulo y de sus dependencias.

4. Volver a generar los datos de las tablas de Asterisk y compararlos con los datos guardados en la fase 2. Sincronizar las diferencias con las tablas de Asterisk y guardar los registros cambiados para ejecutar la acción post-sincronización.

5. Consolidación (*commit*) de la transacción.

6. Con los registros cambiados obtenidos en la fase 4, ejecutar las acciones post-sincronización de las tablas de Asterisk para el módulo y sus dependencias.

7. Ejecutar la acción post-sincronización del módulo y sus dependencias.

En las fases 2 y 4 puede pensarse que no es necesario guardar el estado anterior de las tablas de Asterisk, consultando directamente las tablas para obtener las diferencias. Esto sería posible si la aplicación no trabajase con datos de diferentes empresas. Si consultamos directamente las tablas de Asterisk deberíamos filtrar por empresa para obtener las diferencias. Esto supone un problema ya que cuando cargamos datos en las tablas de Asterisk perdemos la referencia a la empresa, y por tanto la capacidad de filtrar. Además es posible que módulos diferentes inserten datos en la misma tabla de Asterisk. El diagrama de la Fig. 3.8 muestra los pasos descritos anteriormente.

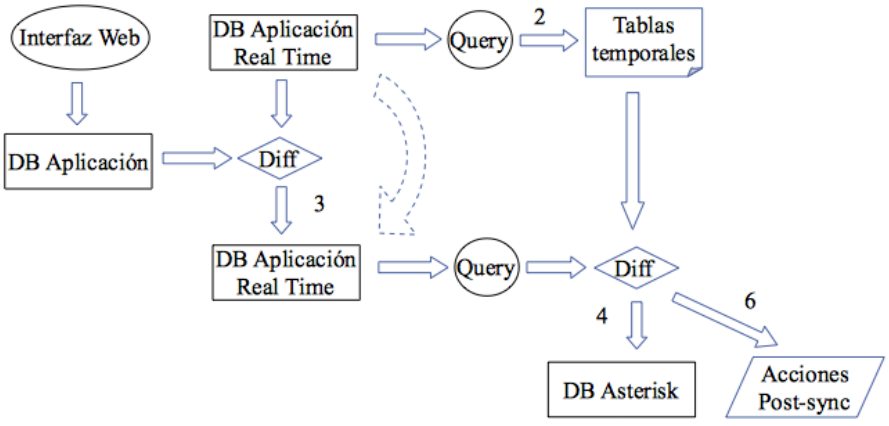


Figura 3.8: Sincronización bases de datos. La base de datos de aplicación tiempo real aparece dos veces, pero representa la misma base de datos en momentos de tiempo distintos.

La sincronización de tablas de la aplicación se realiza mediante el *helper* `SynchronizeTable`, que realiza una sincronización basándose en los campos de auditoría, fecha de creación, usuario etc.

Para la sincronización de las tablas de Asterisk (fase 4) se utilizan los métodos `getChanges` y `syncTable` de la clase `AsteriskModule`. Aquí no es posible utilizar el *helper* `SynchronizeTable` ya que no disponemos de campos de auditoría.

3.4.3. Limitaciones

MySQL no soporta diferir el chequeo de integridad referencial hasta la finalización de la transacción. Para superar esta limitación, las tablas de aplicación que se sincronizan en un módulo se han especificado en un orden que permite que la integridad referencial se mantenga en todo momento. Para ello se elaboraron unas clases *PHP* abstractas en las que se especifican las tablas que componen cada módulo —tanto de la base de datos de la aplicación como de la base de datos Asterisk—, qué módulos dependen del módulo que se está sincronizando y qué acciones o ficheros ejecuta o genera respectivamente, tras la sincronización. De esta manera nos aseguramos de que se sincronicen todas las tablas relacionadas y se mantenga la integridad referencial que nombraba más arriba.

También se ha tenido especial cuidado en no establecer dependencias circulares, ya que no están soportadas por la gestión de dependencias y provocarían el mal funcionamiento de la sincronización.

Para ejecutar las acciones post-sincronización de Asterisk se hace uso de una clase *PHP* llamada `PAMI` (*PHP Asterisk Manager Interface*) que nos permite ejecutar esas acciones de Asterisk facilitando las tareas de mantenimiento y ahorrando mucho tiempo al administrador, ya que no tiene que acceder vía terminal a la centralita, modificar los ficheros necesarios e ir ejecutando cada uno de los comandos post-sincronización cada vez que haya una sincronización de las tablas. La propia aplicación lo ha hecho de manera automática gracias a la integración con `PAMI`.

3.5. Funcionalidades

En esta parte del capítulo se resumen las principales funcionalidades de la aplicación. La descripción completa y detallada se puede consultar en el anexo C - Manual de usuario-.

3.5.1. Widget

En la parte superior derecha del escritorio existe un widget con distintos apartados. La principal función de este Widget es la de mostrar de una manera rápida

y precisa la información que un usuario puede necesitar:

- Cuentas: Son las cuentas que tiene dadas de alta el usuario (el fijo de la mesa, un soft-phone en el ordenador o móvil etc.) Desde este widget podemos ver rápidamente cual es el estado de la cuenta. Si está conectada, desconectada o desconocido.
- Colas: Son las distintas formas en las que puede llegar una llamada. Para que pueda llegar una llamada a través de una cola, el usuario debe estar conectado a la misma. El widget muestra el estado en cada una de las colas.
- Llamadas en cola: Si estamos hablando por teléfono en este apartado aparece la duración de la conversación. Si hay una llamada en espera saldrá debajo.

3.5.2. Mi extensión

Este es el primer icono que aparece en el escritorio. Desde este módulo el usuario podrá cambiar los parámetros básicos de su configuración: contraseña, PIN, opción no molestar (para no recibir llamadas), insertar correo electrónico para recibir alertas de buzón de e-mail o de llamadas perdidas, desviar llamadas, elegir qué ocurre si llega otra llamada durante una conversación.

3.5.3. `Click-to-Call`

Esta función permite efectuar una llamada pulsando sobre un número de teléfono (se despliegan varios y se elige uno si es el caso). Este proceso ahorra mucho tiempo al operador, evitándole teclear cada número o buscarlo en la agenda del terminal (en caso de que estuviese grabado previamente).

3.5.4. Agenda

Permite gestionar (añadir, borrar y eliminar) los contactos. Un contacto puede ser público o privado. Los números marcados como privados no son visibles por otros usuarios. Cada usuario sólo puede gestionar sus propios contactos, es decir, no puede acceder a un contacto público creado por otro usuario y borrarlo. Un contacto puede tener varios números de teléfono y varias direcciones de correo electrónico.

Los grupos de contacto sirven para agrupar varios contactos dentro de un mismo grupo. Un contacto puede pertenecer más de un grupo de contacto. De la misma manera que en los contactos, los grupos también pueden ser públicos o privados.

En la parte superior de la ventana tenemos un buscador que permite encontrar un usuario rápidamente. Se puede filtrar por dos campos:

- Campo buscar: Busca coincidencias tanto en nombre como en apellidos y va filtrando según vamos insertando caracteres.
- Combo grupo: Permite filtrar por grupo de contacto mostrando los que cumplan el texto del campo buscar y pertenezcan a ese grupo.

En cualquier parte de la agenda, podemos llamar a cualquier contacto con tan sólo pulsar encima del número. Esto es posible a la función `Click-to-call` que hemos explicado antes.

3.5.5. Registro de llamadas

Desde este módulo se pueden visualizar todas las llamadas recibidas, realizadas y perdidas. Es muy práctico porque podemos buscar una llamada que teníamos pendiente rápidamente, o un número que nos llamó y que no recordábamos. Además, se puede filtrar por fechas (desde y hasta) y números o nombres a través del buscador.

De la misma manera que la agenda, podemos hacer uso de la función `Click -to-call` para llamar al número deseado.

3.5.6. Buzón de voz

Desde esta ventana se pueden escuchar nuevos mensajes de voz nuevos o volver a escuchar los anteriores. Por defecto y si el navegador lo permite, ejecutará el audio con el codec de *HTML5*. Si no, lo ejecutará mediante el reproductor flash. Los mensajes de buzón de voz se pueden eliminar desde la misma ficha que se abre para escucharlo.

3.5.7. Gestión de usuarios y permisos

Desde este panel se pueden añadir, modificar y eliminar usuarios, roles y grupos de permisos de una empresa.

Usuarios

Este panel es muy parecido al de `Mi extensión`, pero incluye opciones adicionales:

- Cuentas: se pueden añadir, modificar y eliminar cuentas del usuario. Una cuenta define la forma en que un usuario puede llamar y recibir llamadas.
- Roles: se pueden añadir o quitar roles del usuario. Estos roles permiten visualizar o no partes de la aplicación.

- Formas de contacto: cola, extensión etc.
- Permisos de llamada: Se pueden añadir o quitar permisos de llamada al usuario.
- Grupos de captura: Se pueden añadir o quitar grupos de captura al usuario. Con estos grupos, si suena un teléfono de un usaurio A que pertenece al mismo grupo de captura que el usuario B este podrá capturar la llamada del usuario A y atenderla.

Roles

A un rol se le puede añadir o quitar grupos de permisos.

Grupos de permisos

A un grupo de permisos se le puede añadir o quitar permisos.

3.5.8. Configuración de la centralita

Este panel sólo es accesible si el usuario tiene rol administrador. Desde aquí se puede configurar por completo la centralita virtual de empresa. Cuando abrimos la ventana, aparece divida en dos: a la izquierda se sitúan los distintos módulos de configuración y a la derecha se cargará cada uno de los módulos que seleccionemos. De esta manera podremos añadir, modificar y eliminar extensiones, grupos de llamada, colas, permisos de llamada, grupos de captura, contextos, líneas de enlace (*trunks*), rutas de salida, planes, menús de voz, horarios, fabricantes, modelos, equipos, aplicaciones y locuciones.

A lo largo de esta sección resumiremos cada uno de estos aspectos que figuran en el panel de la centralita.

Extensiones

Son las conexiones internas que utilizan los usuarios o terminales de la empresa. En nuestro caso, siempre estará formado por un número, puesto que la gran mayoría de terminales serán fijos y el teclado es del tipo numérico.

Dentro del panel extensión podremos configurar los siguientes campos:

- Número: el número de la extensión.
- Contexto: un contexto de la empresa.
- Grupo de captura: el grupo de captura al que pertenece la extensión (por defecto ninguno).

- Tipo de destino: el tipo de destino de la extensión, que pueden ser usuario, ring group, cola, menú de voz o externa.

- Destino: Tanto la etiqueta de este campo como los valores que contiene, varían en función del tipo de destino seleccionado.

- Plan: un plan de la empresa. Para todos los tipos de destino se trata de un campo autocompletable con el nombre del destino excepto para el tipo de destino externa, que será un campo de texto de tipo numérico para insertar un número de teléfono.

Colas

Una Cola permite establecer de forma totalmente automática un orden secuencial para las llamadas, y también estrategias para su distribución entre los distintos usuarios o agentes que forman parte de la empresa.

Para que un usuario que pertenece a una cola pueda recibir una llamada entrante tiene que haber entrado en sesión. De esta forma si un usuario está ausente de la oficina, su teléfono no sonará cuando entren llamadas a la cola de su departamento.

Grupos de llamada (*Ring groups*)

Son grupos de usuarios que reciben a la vez una llamada. A diferencia de las colas, en un grupo de llamada no es necesario que el usuario esté en sesión para que le suene el teléfono.

Permisos de llamada

Definen los permisos de llamada que posee una empresa. Posteriormente se asignarán a los usuarios desde el panel `Gestión de usuarios y permisos`. Los permisos de llamada está formados por un nombre, una descripción y una expresión regular que define si el número que está marcando el usuario cumple con alguno de sus permisos de llamada.

Grupos de captura

Un grupo de captura sirve para poder atender llamadas de otro usuario que pertenece, por ejemplo, al mismo departamento

Contextos

Un contexto es un punto de entrada del Dialplan. Aquí definimos, por seguridad, el contexto por defecto para todas las conexiones *SIP*, de modo que se rechaza automáticamente toda llamada fuera de cualquier contexto definido.

El **Dialplan**, o plan de marcado, es una colección ordenada de acciones que se ejecutan cuando alguien marca un número dentro de nuestro Asterisk. Un ejemplo trivial es que cuando alguien marca la extensión de otra persona, por ejemplo "3001", suena el teléfono de ese usuario. Sin embargo, se pueden hacer cosas mucho más avanzadas, como por ejemplo gestionar las llamadas en función de un horario, crear una centralita automática de recepción de llamadas, grabar conversaciones, poner música en espera, etc.

Líneas de enlace (Trunks)

Son las que permiten conectar la centralita con la red pública de telefonía. Existen las siguientes opciones:

- Enlaces *RDSI*: interfaces *BRI o PRI*. Existen configuraciones con 2, 4, 6, 8, 10 o 12 *BRI* o 1, 2, 4, 8 o 16 *PRI*. Todos ellos se suministran con cancelación de eco por Hardware.

- Enlaces Analógicos (*FXO*): configuraciones para 1/2/4/8 enlaces, con cancelación eco por Hw.

- Enlaces directos GSM, vía Gateways voip-GSM. Son equipos IP que incorporan ranuras para colocar las tarjetas SIM, y oportan directamente tecnología 2G/3G para conexión con las redes de operador de móvil.

- Enlaces *VoIP*: *SIP, IAX y Skype*. Para optimizar las conexiones *SIP* es recomendable incorporar tarjetas de transcodificación para poder comprimir las sesiones *RTP*. El codec que se utiliza normalmente es el G729.

Rutas de salida

Las rutas de salida llevan un nombre, una descripción, una expresión regular que define la regla que ha de cumplir el número al que queremos llamar y un *DID* por defecto. El *DID (Direct Inward Dialing Number)* es el número que mostraremos al destinatario cuando llamemos por esta ruta de salida y, a su vez, el número que tenemos contratado con los proveedores de telefonía *IP* para que entren las llamadas hacia nuestra centralita.

Planes

Los planes nos sirven para definir estrategias sobre cómo sonarán los teléfonos - dependiendo del horario o no - tanto a nivel de empresa como de usuario. Ejemplo: Llamar a todas las cuentas, llamar sólo a las cuentas fijo y a los diez segundos a las cuentas móvil, etc.

Menús de voz

Los menús de voz sirven para configurar, a través de una serie de estrategias, que ocurre cuando una llamada entra en la centralita. De esta manera, si estamos en horario festivo o cerrado, podemos configurar una locución con el horario de la empresa y, por ejemplo, dejar que dejen un mensaje de voz al finalizar esta locución. Del mismo modo, esta vez en horario de trabajo, podemos configurar una locución de bienvenida en el que se den varias opciones (pueden ser tanto por voz como por teclado) para acceder a algún departamento de la empresa en concreto, o llamar a una extensión, o simplemente no hacer nada. De la misma manera también se pueden configurar estrategias para el modo en que suenan los teléfonos. Podemos configurar que suenen todas las cuentas, todas las cuentas de una cola, las cuentas del tipo fijo y a los diez segundos las cuentas de tipo móvil, etc.

Horarios

En el apartado se definen los distintos horarios de trabajo de la empresa (horario de verano, horario de invierno, etc.) Se pueden definir intervalos de fechas, intervalos de horas y los días que queremos que se aplique el horario, así como, si queremos que este horario se repita cada año.

Fabricantes, Modelos y Equipos

Un fabricante es el nombre del fabricante del terminal, para poder catalogarlo.

Un modelo pertenece a un fabricante y está formado por un nombre y una plantilla con unas variables que son las que harán que el terminal se configure de manera desatendida al pasarle los parámetros adecuados.

Un equipo está formado por una *MAC*, un fabricante, un modelo y un usuario. Con estos datos al conectar el terminal a la red, se conectará a la centralita y se configurará de manera automática con los parámetros que hayamos configurado desde el panel previamente. Esto es útil para poder ofrecer teléfonos a clientes sin tener que desplazarnos a sus oficinas a instalarlo y configurarlo a mano.

Aplicaciones

Las aplicaciones son números con los que se pueden realizar acciones. Estos números y sus acciones vienen predefinidos por Asterisk, pero se pueden configurar a nivel de empresa. Algunas de estas acciones son: desvío de llamadas, captura de llamada de grupo, acceso al buzón personal, etc.

Locuciones

En el apartado locuciones se suben aquellas locuciones personalizadas que queremos que se escuchen tanto para los mensajes de bienvenida, cuando nos llaman a la empresa, como para los buzones de voz.

3.6. Ofuscación

No se han aplicado métodos de ofuscación debido a que este software se instalará en servidores propios. Gracias al framework, que usa un patrón de diseño MVC, es complicado que alguien pueda acceder al código de la aplicación.

No obstante, la empresa dispone de una licencia de *ionCube* [10] - en concreto ionCube PHP Encoder 9.0 Pro - para poder cifrar el código fuente si se decidiera instalarla en un servidor ajeno y asegurarse de que no puedan revender ni utilizar el código.

El software **ionCube PHP Encoder** convierte el código fuente PHP en código binario, haciendo ilegible su lectura y desencriptando en tiempo real cuando se ejecuta desde el navegador. Para el usuario final es completamente transparente, no se nota en tiempo de carga y aporta seguridad sobre el código, sin tener que facilitar un código fuente propietario, sino tan sólo su uso.

Capítulo 4

Conclusiones

Se ha conseguido, en el tiempo esperado, construir una aplicación que nos permite configurar desde cero una centralita virtual para una empresa y ofrecer el servicio en pocos minutos tras disponer de los números de teléfono proporcionados por el proveedor de telefonía (*IP*). Este sistema está operando a pleno rendimiento tanto en la empresa en que se desarrolló, Diaple Networking S.L., como en distintas empresas que han puesto su confianza en los servicios de la compañía.

Está prevista la continuación del proyecto en las siguientes líneas:

- Fax: Incorporación de un módulo de envío y recepción de faxes desde la aplicación.
- Facturación: Inclusión de la facturación de las llamadas desde la aplicación
- Call Center: Mediante campañas la centralita llama automáticamente a clientes y, cuando el cliente descuelga salta una locución y le conecta con un agente de la oficina (ejemplo: una empresa que realice reparaciones de calderas, puede programar campañas por fechas y municipios).

4.1. Conclusiones personales

El desarrollo de este proyecto me ha aportado conocimientos técnicos y experiencia de trabajo en un entorno profesional.

En cuanto a los conocimientos técnicos me gustaría destacar que gracias a la formación recibida durante la realización del proyecto he llegado a comprender mucho mejor como trabajar en proyectos de software de gran envergadura y abarcarlos con metodologías de Ingeniería del software.

He aprendido a trabajar con *JavaScript/AJAX* [1, 2], uso de *JSON* [11], uso de bibliotecas como *JQuery* —que facilitan y aceleran la programación— y he

ampliado mis conocimientos sobre *PHP, HTML y CSS3*. Aunque ya los había utilizado con anterioridad, no había necesitado hacer un uso tan exhaustivo de la documentación. La utilización de clases y métodos que no conocía me ha dado soltura en la consulta eficiente de la documentación disponible. Quiero añadir que me parecen lenguajes de gran importancia para la vida profesional de un ingeniero de software en la actualidad y que apenas se toca en algunas optativas —pueden no ser cursadas— durante la formación académica de la titulación de Ingeniería Informática.

Este proyecto me ha aportado habilidad en el trabajo en equipo bajo la dirección de un director técnico, metodología en la consulta y reutilización de trabajos previos de otros compañeros, formación en el entorno de desarrollo de la empresa, y experiencia en pruebas y depuración del sistema y en su posterior implantación y explotación.

La magnitud del proyecto me ha hecho ver también la importancia de aspectos que hasta ahora, en proyectos de menor calado, me habían pasado desapercibidos, como por ejemplo la imprescindible organización de los proyectos, la necesidad de una buena metodología, la realización de pruebas o la importancia del diseño frente a la implementación.

A nivel personal estoy muy satisfecho, el trabajo ha terminado en el tiempo estimado y está funcionando perfectamente. El ambiente de trabajo ha hecho que el proyecto haya constituido una experiencia de aprendizaje no sólo eficaz sino además agradable.

Lo más gratificante ha sido que tanto jefes como el equipo han quedado contentos con mi trabajo e implicación hasta el punto de que me han ofrecido entrar a formar parte de la plantilla de la empresa.

Bibliografía

[1] JavaScript: http://www.w3schools.com/js/.

[2] AJAX: http://searchwindevelopment.techtarget.com/tutorial/Ajax-Learning-Guide.

[3] Trygve Reenskaug. MODELS - VIEWS – CONTROLLERS. Technical report, Xerox PARC, 1978.

[4] HTML5: http://www.w3schools.com/html/.

[5] CSS3: http://www.w3schools.com/css/css3/.

[6] jQuery: http://jquery.com/.

[7] jQuery UI: http://jqueryui.com/.

[8] MySQL: http://dev.mysql.com/doc/refman/5.7/en/index.html.

[9] PHP: http://www.php.net/manual/es/.

[10] ionCube PHP Encoder: http://www.ioncube.com/php_encoder.php/.

[11] JSON: http://docs.1060.org/docs/3.1.0/book/discovered/doc_mod_json_guide.html.

Apéndice A

Planificación y control de esfuerzos

A.1. Planificación

Inicialmente se calculó la distribución de tiempos de cada fase del desarrollo ajustando la misma a 500 horas de trabajo, al final se tuvieron que ajustar estos tiempos para añadir tiempo en corrección de errores o afinar detalles al poner en producción el sistema y en el desarrollo de la documentación.

En la figura A.1 se muestra el diagrama de gantt que representa la distribución final de las fases del proyecto. En el mes de enero se realizaron menos horas coincidiendo con el periodo de exámenes.

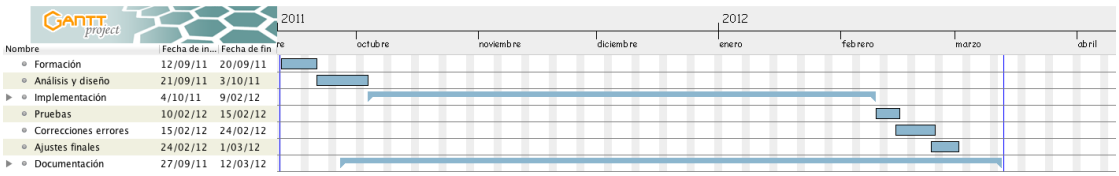


Figura A.1: Diagrama de la distribución temporal de las fases del proyecto

A.2. Control de esfuerzos

Durante todo el proyecto se han ido apuntando en una hoja de cálculo las horas que se iban dedicando a cada uno de los apartados del proyecto (análisis y diseño, implementación, etc.) así como una breve descripción de qué había hecho con más detalle ese día. Las tareas realizadas se han agrupado posteriormente en categorías que se describen a continuación:

- **Formación:** Horas invertidas en formación e investigación de nuevas tecnologías o metodologías de trabajo.
- **Reunión:** Horas dedicadas a reuniones con el director del proyecto o compañeros de la empresa. Normalmente una cada dos semanas con una duración estimada de treinta minutos.
- **Análisis y diseño:** Horas invertidas en el proceso de análisis y diseño de los componentes del proyecto.
- **Implementación:** Horas dedicadas a la programación de la aplicación.
- **Pruebas:** Horas dedicadas a la comprobación del correcto funcionamiento de la aplicación así como su depuración, estas pruebas corresponden con las pruebas finales de integración de todos los módulos una vez puesta en fase de pruebas. Durante la fase de implementación se fueron realizando paralelamente pruebas unitarias y funcionales, las cuales están contabilizadas en las horas de implementación.
- **Corrección de errores:** Horas dedicadas a la corrección de errores encontrados en la aplicación durante el periodo de pruebas.
- **Ajustes finales:** Horas dedicadas a modificaciones, mayoritariamente de visualización, una vez terminada la aplicación.
- **Documentación:** Horas invertidas en la redacción de documentos tanto internos (manual de usuario, manual técnico, wiki de la empresa) como esta memoria.

El desarrollo del sistema continúa tras la realización de este proyecto, por una parte se introducirán los puntos nombrados en líneas futuras de la memoria, en el capítulo 4 - Conclusiones -, y por otra parte se reciben nuevas necesidades/mejoras por parte de los clientes, se evalúan y si se considera apropiado se desarrollan.

Tarea	Horas
Formación	35
Reuniones	10
Análisis y diseño	35
Implementación	315
Pruebas	20
Corrección de errores	15
Ajustes finales	10
Documentación	110

Tabla A.1: Horas empleadas en cada tarea

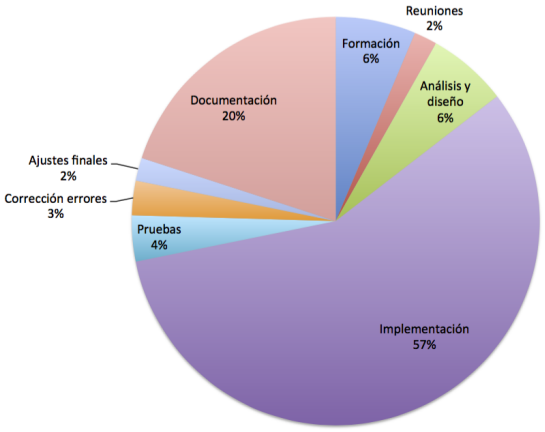


Figura A.2: Gráfico distribución de tareas por horas

Mes	Horas
Septiembre	70
Octubre	90
Noviembre	100
Diciembre	80
Enero	70
Febrero	105
Marzo	35
Total	560

Tabla A.2: Horas empleadas por mes

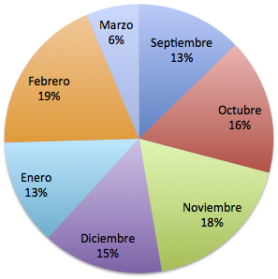


Figura A.3: Gráfico distribución de esfuerzos

Este proyecto se ha realizado en 6 meses desde Septiembre 2011 hasta Marzo 2012.

Puesto que se ha combinado la realización de este proyecto con el desarrollo de asignaturas de la universidad, la dedicación del proyecto ha sido de media jornada (con una media de 5 horas diarias) y con una ausencia de casi dos semanas en periodo de exámenes. Sumando las horas totales aplicadas en cada mes, se obtienen las horas totales dedicadas a la realización del proyecto dando un total de 560 horas.

Apéndice B

Análisis y diseño

B.1. Análisis de requisitos

A continuación se exponen los requisitos funcionales que debe cumplir la aplicación, organizadas por módulos:

Aplicación Web:

- Acceso restringido mediante usuario y contraseña

Super-administrador:

- El super-usuario puede añadir, modificar y eliminar empresas y DID's contratados (números de teléfono)
- El super-usuario puede navegar entre todas la empresas y puede saltar de una a una desde el pantalla principal.
- El super-usuario posee todos los permisos de la aplicación.
- El super-usuario tendrá acceso a una ventana con un terminal linux.

Controles de usuario:

- Configurar extensión:
  - Podrá modificar la contraseña de acceso.
  - Podrá modificar el pin de seguridad que se utilizará para permisos de ciertas acciones desde el teléfono.
  - Podrá configurar la opción de no molestar desde el frontal.
  - Podrá enviar alertas por e-mail (definiendo a que e-mail lo quiere enviar):

        - Buzón de voz
        - Llamadas perdidas
    - Podrá desviar llamadas:
        - A un número
        - A un buzón
    - Podrá definir acciones mientras habla:
        - Permitir rellamada
        - Recibir llamadas en segundo plano.
    - Podrá visualizar las formas de contacto asociadas.
    - Además en las de tipo usuario podrá definir el horario en que recibirá las llamadas.

- Agenda:
    - Podrá registrar contactos privados o públicos.
    - Podrá acceder a contactos creados por otros usuarios con carácter público
    - Podrá marcar como favoritos aquellos contactos que utilice más a menudo.
    - Podrá crear grupos de contactos y asociarlos con contactos.
    - Podrá buscar contactos suyos y públicos de otros usuarios.
    - Podrá llamar pinchando sobre el número de teléfono.
        - Además, si tiene más de una forma de contacto el usuario, elegirá con cual llamar.
    - Los contactos podrán tener uno o más teléfonos.
    - Los contactos podrán tener uno o más e-mails.
    - Los contactos podrán pertenecer a uno o más grupos de contactos.
- Registro de llamadas:
    - Podrá visualizar las llamadas entrantes.
    - Podrá visualizar las llamadas saliente.
    - Podrá visualizar las llamadas perdidas.
    - Podrá filtrar las llamadas entre fechas (desde y hasta).
    - Podrá filtrar por llamadas:

        - "Mis llamadas"
        - "Llamadas a la empresa"
    - Podrá llamar a cualquiera de los números (entrantes, salientes o perdidas) pinchando sobre el número:
        - Si tiene más de una forma de contacto el usuario, elegirá con cual llamar.

- Buzón de voz:
    - Se dividirán en pendientes y escuchados
    - Podrá escuchar tantas veces como quiera un buzón de voz.
    - Cuando se escucha un buzón pendiente pasa al apartado de escuchados.

Controles de administrador

- Gestión de usuarios y permisos
    - Usuarios
        - Se podrán visualizar los usuarios de la empresa(sólo una a la vez).
        - Se podrá agregar, modificar y eliminar usuarios de la aplicación (dentro de su empresa)
        - Se podrán consultar formas de contacto de un usuario.
        - Se podrá añadir, modificar o eliminar formas de contacto de un usuario.
        - Se podrá definir una extensión por defecto para el usuario.
        - Roles
            - Se podrá añadir o quitar roles a un usuario.
        - Formas de contacto
            - Se muestran las formas de contacto del usuario (tipo cola y usuario)
            - En las de tipo cola se podrán modificar el horario
        - Permisos de llamada
            - Se podrá añadir y quitar permisos de llamada
        - Grupos de captura
            - Se podrá añadir y quitar grupos de captura.
    - Roles
        - Se podrán crear, modificar y eliminar roles.
        - Se podrán añadir y quitar grupos de permisos a un rol.

- Grupos de permisos
    - Se podrán crear, modificar y eliminar grupos de permisos.
    - Se podrán añadir y quitar permisos a un grupo de permisos.

- Configuración de la centralita
    - Extensiones
        - Se podrá visualizar las extensiones de una empresa(sólo la actual)
        - Se podrán añadir, modificar y eliminar extensiones a una empresa.
    - Ring groups
        - Se podrán visualizar los ring groups de una empresa.
        - Se podrán añadir, modificar y eliminar ring groups de una empresa.
    - Colas
        - Se podrá visualizar las colas de una empresa(sólo la actual)
        - Se podrán añadir, modificar y eliminar colas a una empresa
    - Permisos de llamada
        - Se podrán visualizar los permisos de llamada de una empresa.
        - Se podrán añadir, modificar y eliminar permisos de llamada de una empresa.
    - Grupos de captura
        - Se podrán visualizar los grupos de captura de una empresa.
        - Se podrán añadir, modificar y eliminar grupos de captura de una empresa.
    - Contextos
        - Se podrán visualizar los contextos de una empresa.
        - Se podrán añadir, modificar y eliminar contextos de una empresa.
    - Trunks
        - Se podrán visualizar los trunks de una empresa.
        - Se podrán añadir, modificar y eliminar trunks de una empresa.
    - Rutas de salida
        - Se podrán visualizar las rutas de salida de una empresa.
        - Se podrán añadir, modificar y eliminar rutas de salida de una empresa.
    - Planes

    - Se podrán visualizar los planes de una empresa.
    - Se podrán añadir, modificar y eliminar planes de una empresa.
- Menús de voz
    - Se podrán visualizar los menús de voz de una empresa.
    - Se podrán añadir, modificar y eliminar menús de voz de una empresa.
- Horarios
    - Se podrán visualizar los horarios de una empresa.
    - Se podrán añadir, modificar y eliminar horarios de una empresa.
- Fabricantes
    - Se podrán visualizar los fabricantes de teléfonos de una empresa.
    - Se podrán añadir, modificar y eliminar fabricantes de teléfonos de una empresa.
- Modelos
    - Se podrán visualizar los modelos de teléfonos de una empresa.
    - Se podrán añadir, modificar y eliminar modelos de teléfonos de una empresa.
- Equipos
    - Se podrán visualizar los equipos telefónicos de una empresa.
    - Se podrán añadir, modificar y eliminar equipos telefónicos de una empresa.
- Aplicaciones
    - Se podrán visualizar las aplicaciones disponibles de asterisk y sus códigos (login, capturar, etc.).
    - Se podrán modificar los códigos de las aplicaciones de una empresa (se podrán personalizar).
- Locuciones
    - Se podrán visualizar las locuciones de una empresa.
    - Se podrán añadir y eliminar locuciones de una empresa (tipo de archivo de audio: mp3 o wav).

B.2. Diseño de la Base de Datos

En este apartado se explica el diseño de la base de datos de la aplicación, que es la misma que la del realtime tal y como se explica en la memoria y en el apartado de sincronización del anexo D -Manual técnico-.

Para ofrecer una visión general del diseño de esta base de datos, se muestra un diagrama E/R repartido en distintas secciones y explicando las entidades debajo.

A fin de soportar la gestión de múltiples empresas, todas las entidades de la base de datos están relacionadas con la entidad 'Company'. Debido a que esta entidad se relaciona con gran cantidad de líneas no se muestran en los diagramas siguientes las relaciones con esta entidad.

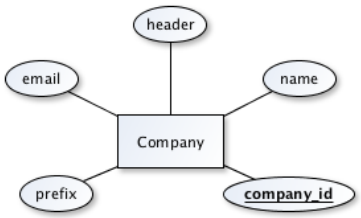


Figura B.1: Entidad Company

Entidad correspondiente a la Fig. B.1:

- Company: (company_id: int; name: cadena, NO NULO; header: cadena; email: cadena; prefix: cadena, NO NULO; admin_user_id: int, clave ajena de User).

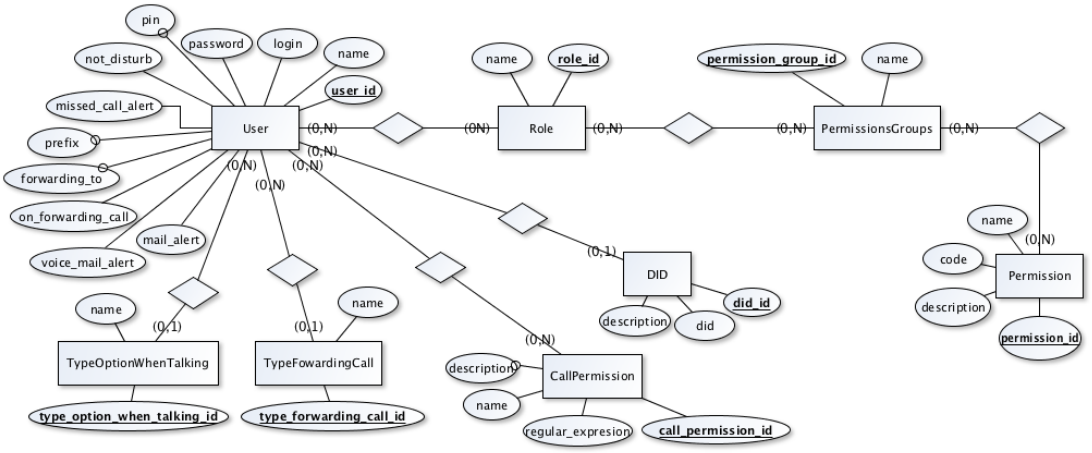


Figura B.2: Usuarios, roles y permisos

Entidades correspondientes a la Fig. B.2:

- Permission: (permission_id: int; code: cadena, NO NULO; name: cadena, NO NULO; description: cadena, NO NULO).
- PermissionGroup: (permission_group_id: int; name: cadena, NO NULO; company_id: int, clave ajena de Company).
- Role: (role_id: int; name: cadena, NO NULO; company_id: int, clave ajena de Company).
- User: (user_id: int; name: cadena, NO NULO; login: cadena, NO NULO; password: cadena, NO NULO; pin: cadena; mail_alert: cadena; voice_mail_alert: boolean, NO NULO; missed_call_alert: boolean, NO NULO; not_disturb: boolean, NO NULO; on_forwarding_call: boolean, NO NULO; prefix: cadena; forwarding_to: cadena; type_forwarding_call_id: int, clave ajena de TypeForwardingCall; type_option_when_talking_id: int, clave ajena de TypeOptionWhenTalking; did_id: int, clave ajena de DID; company_id: int, clave ajena de Company).
- DID: (did_id: int; did: cadena, NO NULO; description: cadena; company_id: int, clave ajena de Company)
- TypeForwardingCall : (type_forwarding_call_id: int; name: cadena, NO NULO)
- TypeOptionWhenTalking : (type_option_when_talking_id: int; name: cadena, NO NULO)
- CallPermission : (call_permission_id: int; name: cadena, NO NULO; regular_expresion: cadena, NO NULO; description: cadena; company_id: int, clave ajena de Company)

Entidades correspondientes a la Fig. B.3:

- Account: (account_id: int; name: cadena, NO NULO; block: booleano, NO NULO; account_type_id: int, clave ajena de AccountType; context_id: int, clave ajena de Context; user_id: int, calve ajena de User; device_id: int clave ajena de Device; technology_type_id: int, clave ajena de TechnologyType)
- AccountType: (account_type_id: int; name: cadena, NO NULO; description:cadena)
- AccountParameter: (account_parameter_id: int; text_value: cadena; number_value:int; date_value: fecha; flag_value: cadena; account_id: int, clave ajena de Account; technology_type_parameter_id: int, clave ajena de TechnolgyTypeParameter)

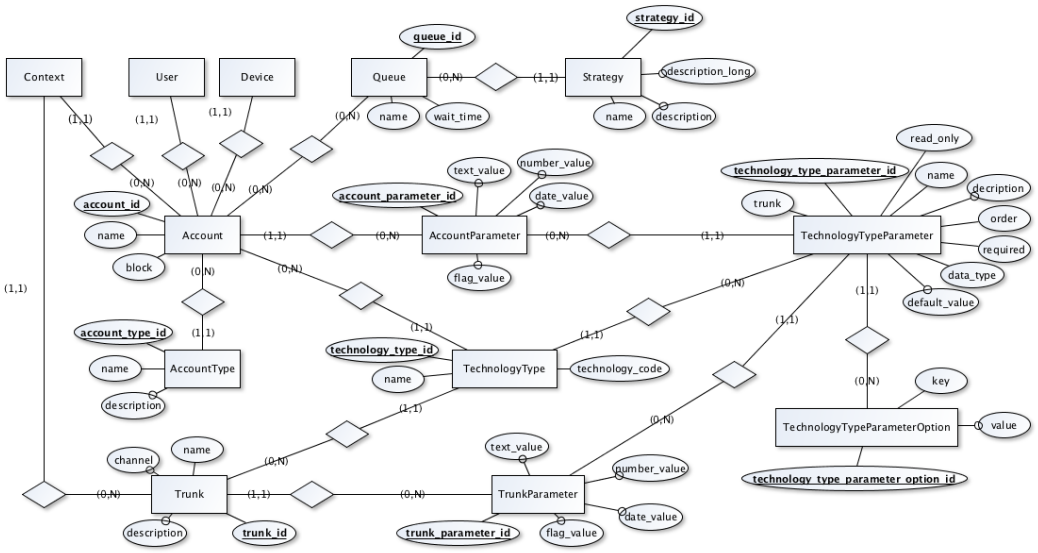


Figura B.3: Formas de contacto

- TechnolgyTypeParameter: (technology_type_parameter_id: int; name: cadena, NO NULO; description: cadena; order: int, NO NULO; required: boolean, NO NULO; data_type: cadena, NO NULO; default_value: cadena; readonly: boolean, NO NULO; technology_type_id: int, clave ajena de TechnologyType)
- TechnolgyType: (technology_type_id: int; name: cadena, NO NULO; technology_code: cadena, NO NULO)
- TechnologyTypeParameterOption: (technology_type_parameter_option_id: int; key: cadena, NO NULO; value: cadena; technology_type_parameter_id: int, clave ajena de TechnolgyTypeParameter)
- TrunkParameter: trunk_parameter_id: int; text_value: cadena; number_value:int; date_value: fecha; flag_value: cadena; technology_type_parameter_id: int, clave ajena de TechnolgyTypeParameter; trunk_id: int, clave ajena de Trunk)
- Trunk: (trunk_id: int; name: cadena, NO NULO; description: cadena; channel: cadena; technology_type_id: int, clave ajena de TechnologyType; context_id: int, clave ajena de Context; company_id: int, clave ajena de Company)
- Queue: (queue_id: int; name: cadena, NO NULO; wait_time: int, NO NULO, strategy_id: int, clave ajena de Strategy; company_id: int, clave ajena

de Company)

- Strategy: (strategy_id: int; name: cadena, NO NULO; description: cadena, description_long: cadena)

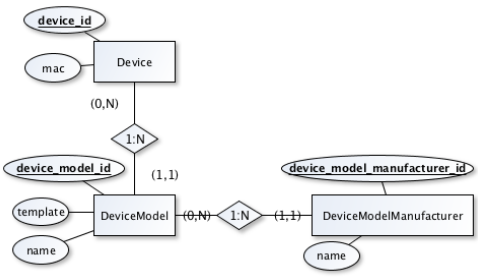


Figura B.4: Dispositivos

Entidades correspondientes a la Fig. B.4:

- Device: (device_id: int; mac: cadena, NO NULO; device_model_id: int, clave ajena de DeviceModel)
- DeviceModel: (device_model_id: int; name: cadena, NO NULO; template: cadena, NO NULO; device_model_manufacturer_id: int, clave ajena de DeviceModelManufacturer)
- DeviceModelManufacturer: (device_model_manufacturer_id: int; name: cadena, NO NULO; company_id: int, clave ajena de Company)

Entidades correspondientes a la Fig. B.5:

- Ivr: (ivr_id: int; name: cadena, NO NULO; description: cadena; comany_id: int, clave ajena de Company)
- IvrGroup: (ivr_group_id: int; name: cadena, NO NULO; description: cadena; comany_id: int, clave ajena de Company)
- IvrItem: (ivr_item_id: int; order: int, NO NULO; exten: cadena, NO NULO; ivr_id: int, clave ajena de Ivr; ivr_item_type_id: int, clave ajena de IvrItemType; ivr_item_option_type_id: int, clave ajena de IvrItemOptionType)

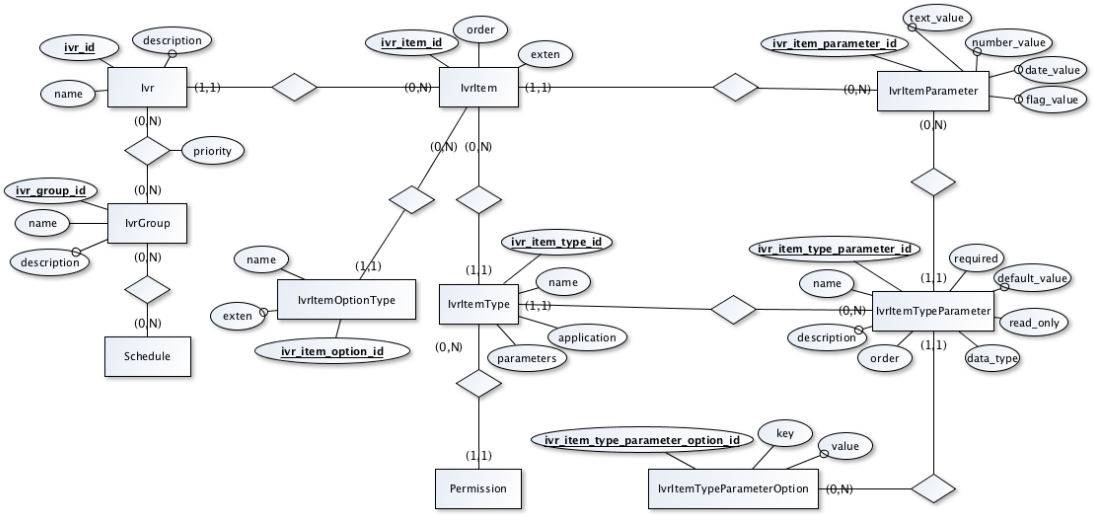


Figura B.5: IVRs

- IvrItemType: (ivr_item_type_id: int; name: cadena, NO NULO; application: cadena, NO NULO; parameters: cadena, NO NULO; permission_id: int, clave ajena de Permission)
- IvrItemOptionType: (ivr_item_option_type_id: int; name: cadena, NO NULO; exten: cadena)
- IvrItemParameter: (ivr_item_parameter_id: int; text_value: cadena; number_value:int; date_value: fecha; flag_value: cadena; ivr_item_type_parameter_id: int, clave ajena de IvrItemTypeParameter; ivr_item_id: int, clave ajena de IvrItem)
- IvrItemTypeParameter: (ivr_item_type_parameter_id: int; name: cadena, NO NULO; description: cadena; order: int; required: booleano, NO NULO; default_value: cadena; data_type:int; readonly: booleano, NO NULO;ivr_item_type_id: int, clave ajena de IvrItemType)
- IvrItemTypeParameterOption: (ivr_item_type_parameter_option_id: int; key: cadena, NO NULO; value: cadena; ivr_item_type_parameter_id: int, clave ajena de IvrItemTypeParameter)

Entidades correspondientes a la Fig. B.6:

- Schedule: (schedule_id: int; name: cadena, NO NULO; start_date: date; end_date: date; start_time: time; end_time: time; private: boolean; mon-

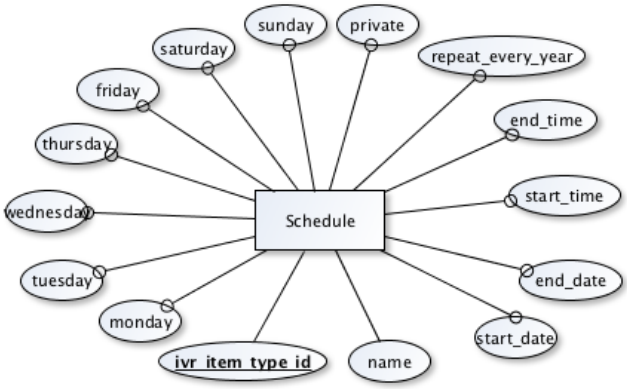


Figura B.6: Entidad horario

day: boolean; tuesday: boolean; wednesday: boolean; thursday: boolean; friday: boolean; saturday: boolean; sunday: boolean; repeat_every_year: boolean; compnany_id: int, clave ajena de Company)

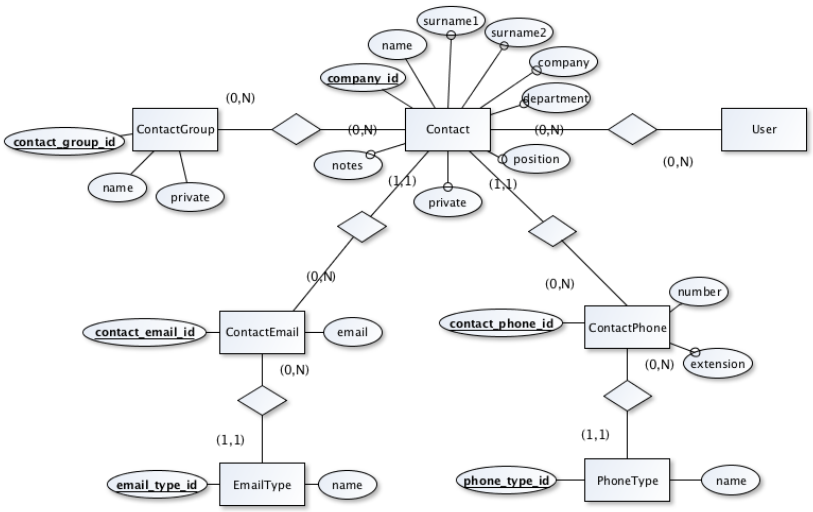


Figura B.7: Contactos

Entidades correspondientes a la Fig. B.7:

- Contact: (contact_id: int; name: cadena, NO NULO; surname1: cadena; surname2: cadena; company: cadena; department: cadena; position: cadena; private: boolean; notes: cadena; company_id: int, clave ajena de Company)

- ContactEmail: (contact_email_id: int; email: cadena, NO NULO; email_type_id: int, clave ajena de EmailType; contact_id: int, clave ajena de Contact)
- EmailType: (email_type_id: int; name: cadena, NO NULO);
- ContactPhone: (contact_phone_id: int; number: cadena, NO NULO; extension: cadena; phone_type_id: int, clave ajena de PhoneType; contact_id: int, clave ajena de Contact)
- PhoneType: (phone_type_id: int; name: cadena, NO NULO);
- ContactGroup: (contact_group_id: int; name: cadena, NO NULO; private: booleano, NO NULO)

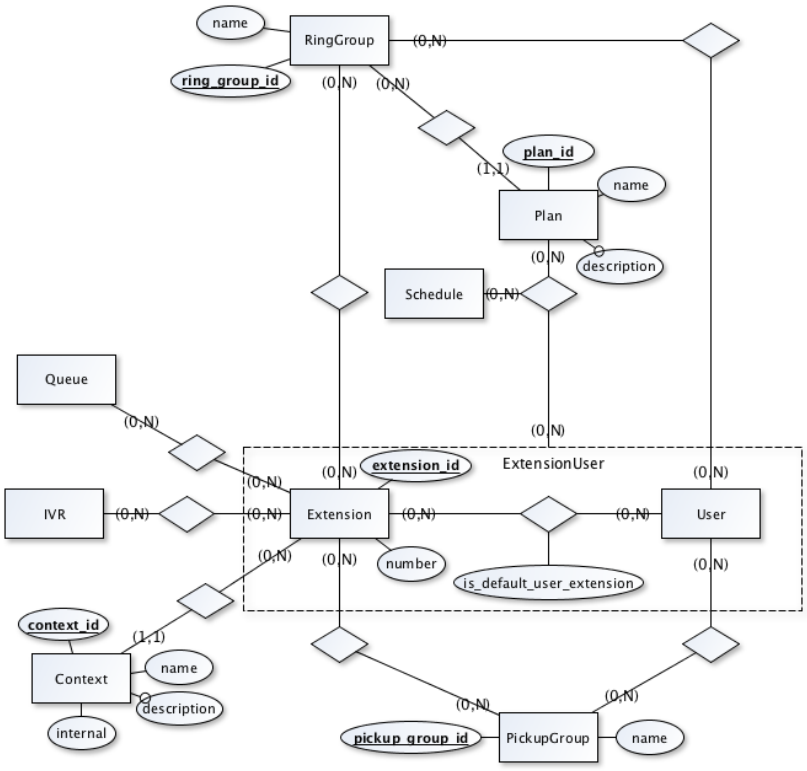


Figura B.8: Extensiones

Entidades correspondientes a la Fig. B.8:

- Extension: (extension_id: int; number: cadena, NO NULO; context_id: int, clave ajena de Context)

- Context: (context_id: int; name: cadena, NO NULO; internal: booleano, NO NULO; description: cadena; company_id: int, clave ajena de Company)
- Plan: (plan_id: int; name: cadena, NO NULO; description: cadena; company_id: int, clave ajena de Company)
- RingGroup: (ring_group_id: int; name: cadena, NO NULO; plan_id: int, clave ajena de Plan; company_id: int, clave ajena de Company)
- PickupGroup: (pickup_group_id: int; name: cadena, NO NULO; company_id: int, clave ajena de Company)

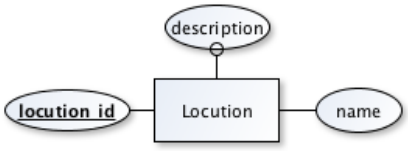


Figura B.9: Locución

Entidad correspondiente a la Fig. B.9:

- Locution: (locution_id: int; name: cadena, NO NULO; decription: cadena; company_id: int, clave ajena Company)

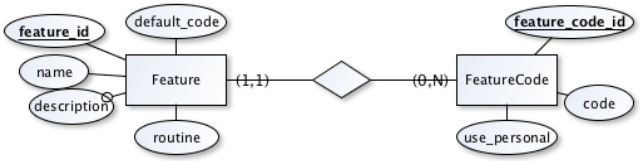


Figura B.10: Características

Entidades correspondientes a la Fig. B.10:

- FeatureCode: (feature_code_id: int; code: cadena, NO NULO; use_personal: booleano, NO NULO; feature_id: int, clave ajena de Feature; company_id: int, clave ajena Company)
- Feature: (feature_id: int; name: cadena, NO NULO; description: cadena; default_code: cadena, NO NULO; routine: cadena, NO NULO)

B.3. Diagrama de flujo de datos

En este apartado se muestran los diagramas de flujos de datos del módulo desarrollado.

En el nivel 0 — Fig. B.11 — se muestran los elementos con los que hemos trabajado en el desarrollo del presente PFC que interactúan con el sistema de la centralita. El usuario se ha dividido en tres tipos - usuario, administrador y super-administrador - para diferenciar las tareas que puede hacer dependiendo del rol que desempeñen.

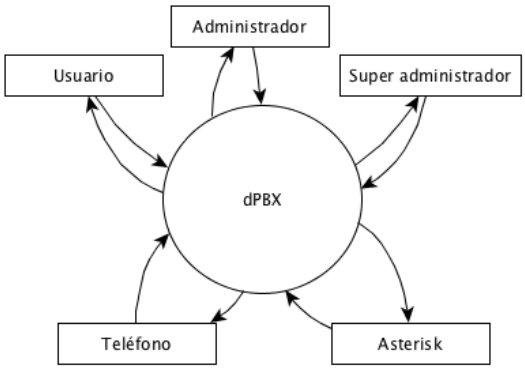


Figura B.11: DFD nivel 0

En la Fig. B.12 se define el nivel 1 en que se describen los diferentes módulos contenidos en subsistema de información.

Estos módulos son los encargados de modificación y alteración de la información:

- **Módulo mi extensión:** Datos de configuración del usuario.
- **Módulo agenda:** Permite acceder a los contactos del usuario.
- **Módulo registro de llamadas:** Permite visualizar las llamadas realizadas, llamadas recibidas y llamadas perdidas del usuario.
- **Módulo buzón de voz:** Permite escuchar mensajes de voz recibidos a la cuenta del usuario.
- **Módulo configuración de centralita:** Permite configurar todos los parámetros internos del funcionamiento de la centralita.

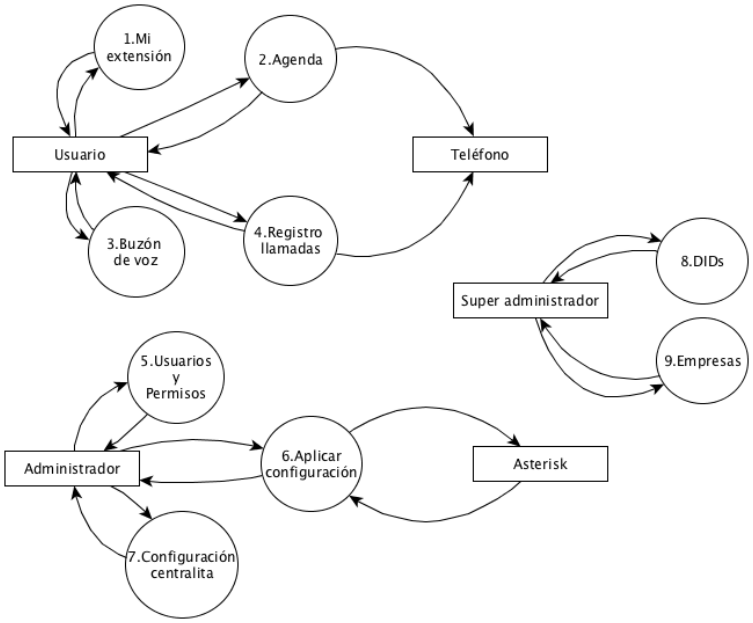


Figura B.12: DFD nivel 1

- **Módulo aplicar cambios:** Permite aplicar los cambios realizados en la configuración para que surjan efecto en la base de datos de tiempo real y en Asterisk. Es decir, para que tengan efecto real sobre la centralita.
- **Módulo DDIs:** Visualizar, dar de alta o modificar los números contratados con el proveedor de telefonía IP.
- **Módulo Empresas:** Visualizar, dar de alta o modificar las empresas de la centralita. Es decir, las centralitas virtuales de cada empresa que están configuradas.

En las siguientes figuras — Fig. B.13, Fig. B.14 y Fig. B.15 — mostramos solamente el diagrama de flujo de datos de nivel 2 de los módulos **Aplicar cambios**, **Registro de llamadas** y **Agenda** para que quede un poco más claro cómo interactúa con Asterisk y Teléfono, ya que en el nivel 1 no termina de quedar claro.

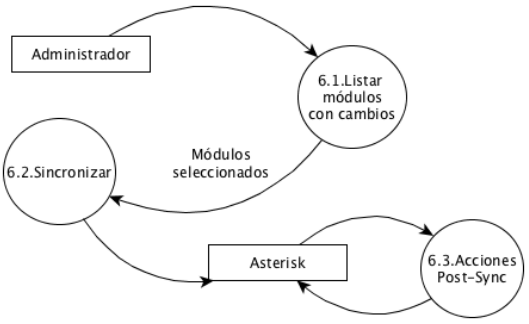


Figura B.13: DFD nivel 2 - Aplicar cambios

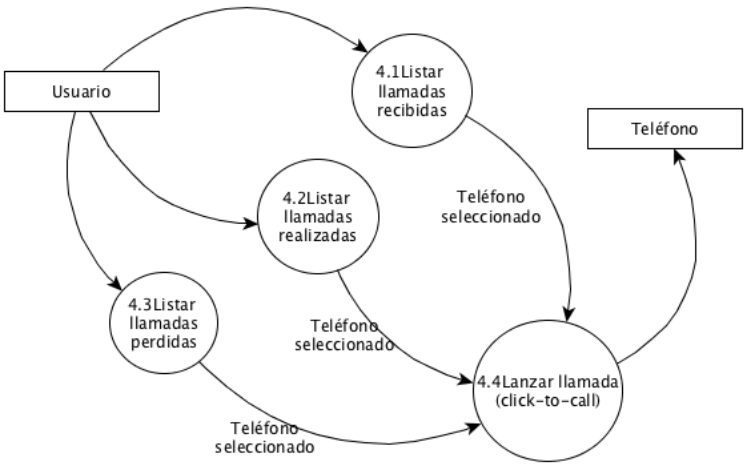


Figura B.14: DFD nivel 2 - Registro de llamadas

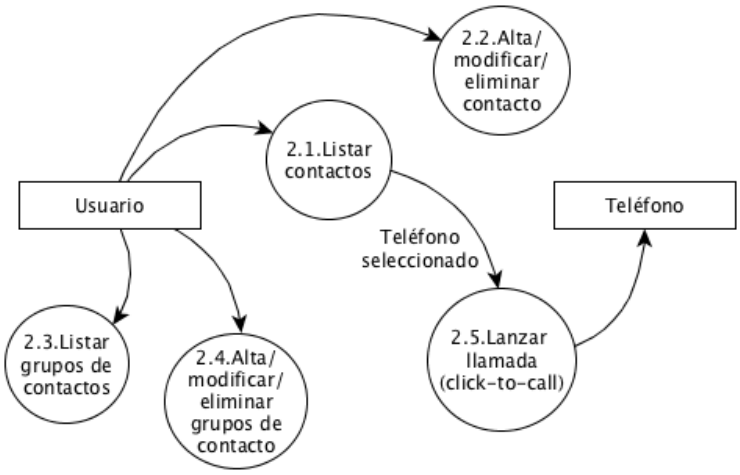


Figura B.15: DFD nivel 2 - Agenda

B.4. Diagrama de navegación

Antes de proceder a diseñar el prototipado se decidieron las vistas que se crearían para interactuar con el módulo desarrollado.

En la Fig. B.16 se muestra el diagrama de navegación en el que se subdivide en dos zonas, la zona de administración diseñada par permitir a los administradores realizar las operaciones que tienen que ver con la gestión de la centralita y la zona de usuario, que es a la que podrán acceder los usuarios para hacer consultas o modificar sus datos.

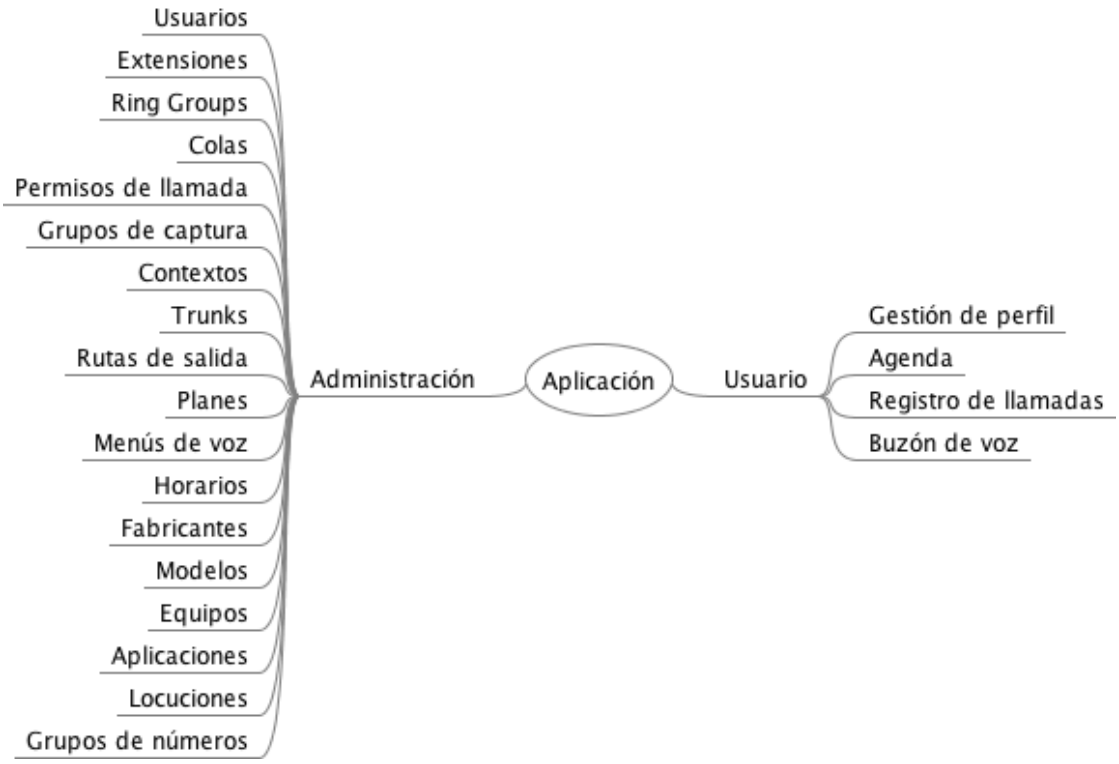


Figura B.16: Diagrama navegación

B.5. Prototipado de ventanas

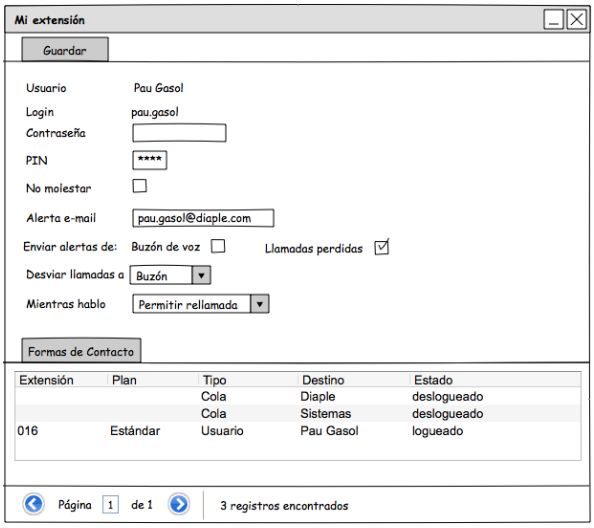


Figura B.17: Mi extensión

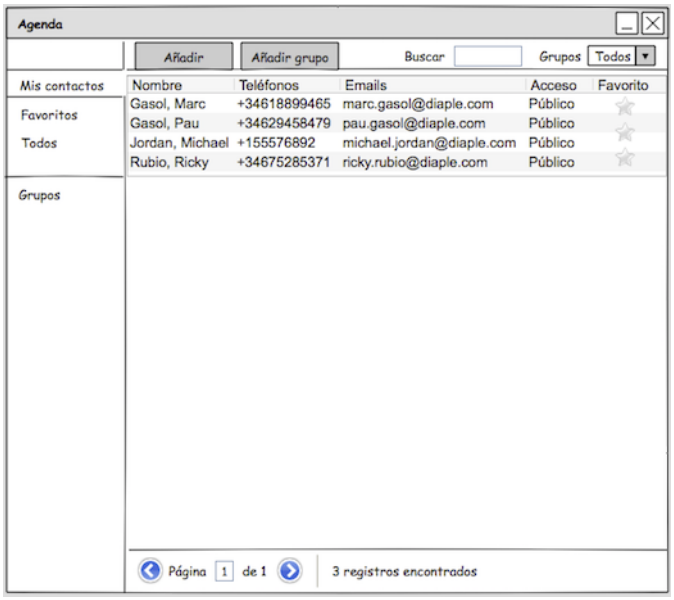


Figura B.18: Agenda

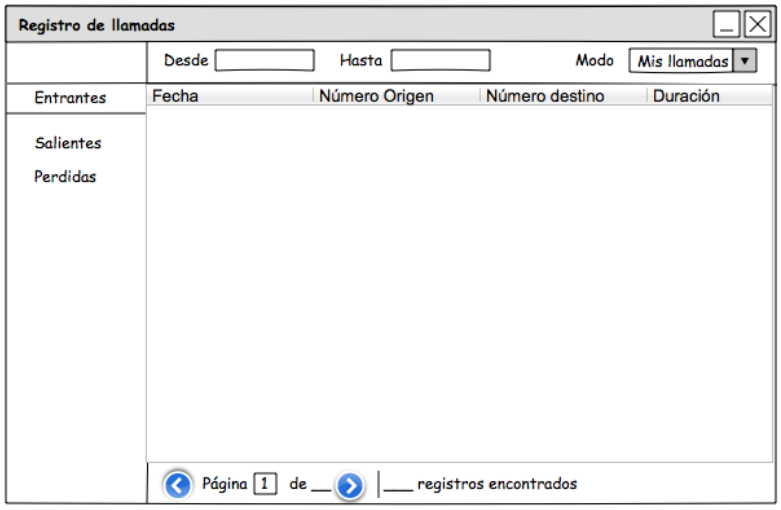


Figura B.19: Registro de llamadas

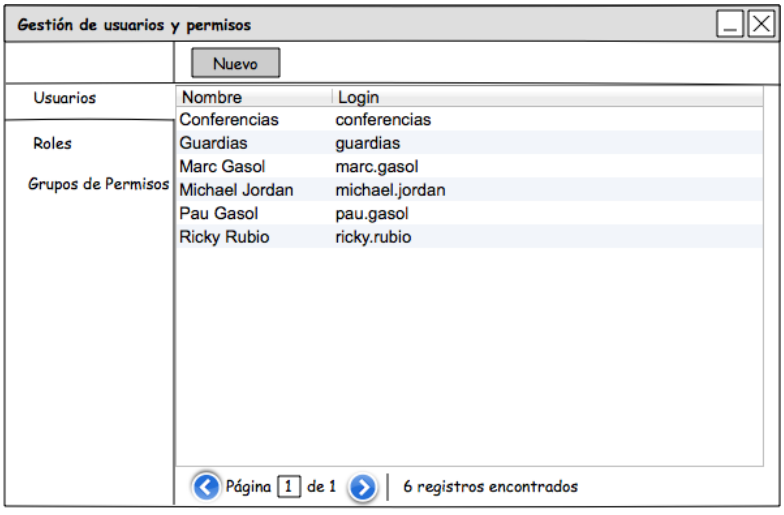


Figura B.20: Gestión de usuarios y permisos

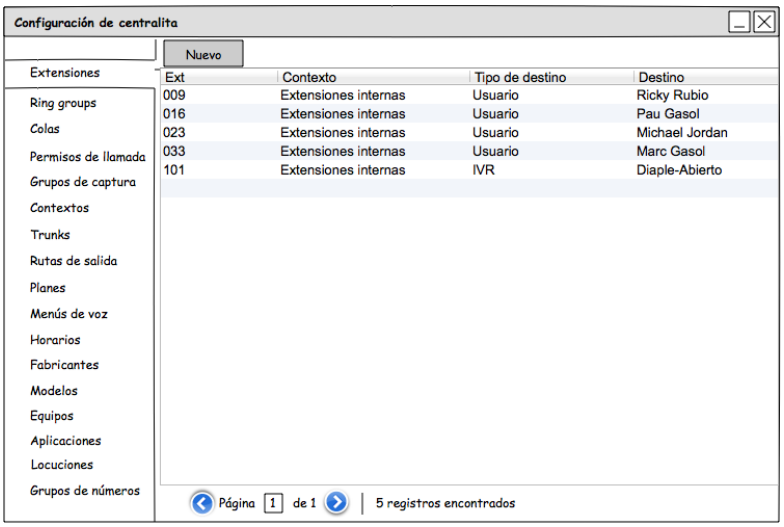


Figura B.21: Configuración centralita

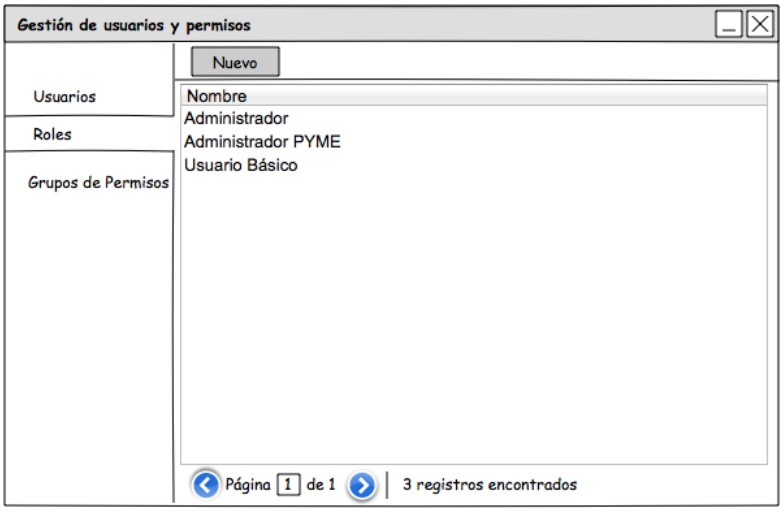


Figura B.22: Roles

Apéndice C

Manual de usuario

C.1. Introdución

La finalidad de la aplicación para un usuario es facilitar el acceso a determinadas herramientas y funcionalidades difíciles de manejar desde un teléfono (y algunas funcionalidades extra de la centralita que no se pueden acceder de otra manera). Aparte de añadir facilidad de manejo y visualización, se añaden otras opciones como alertas por email o acceso a una Agenda muy completa. El menú de la aplicación para un usuario básico queda como mostramos en la figura C.1.

Figura C.1: Visualización iconos usuario básico

A lo largo de este documento vamos a repasar las funciones básicas que puede realizar un usuario paso a paso. Para que en pocos minutos un usuario sepa utilizar al 100 la aplicación.

- **Mi extensión:** El usuario puede modificar sus parámetros.
- **Agenda:** El usuario puede consultar tanto sus contactos privados como los públicos para la empresa.

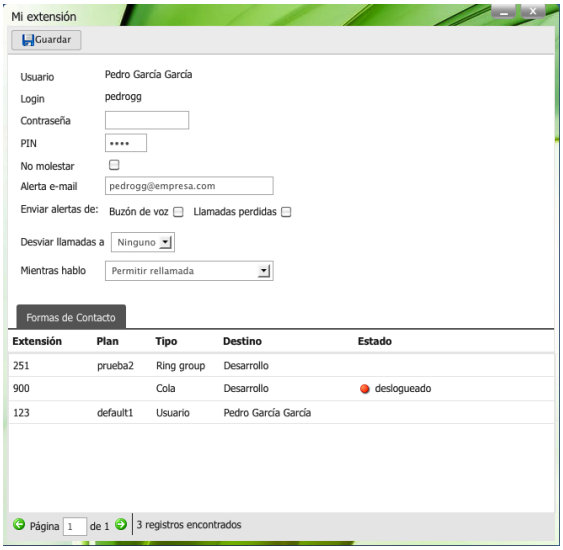
- **Registro de llamadas:** El usuario puede visualizar el histórico de llamadas entrantes, salientes y perdidas.

- **Buzón de voz:** El usuario puede escuchar los mensajes que tiene en el buzón de voz.

C.2. Menús

C.2.1. Acceso a mi extensión

Desde el escritorio podemos acceder a mi extensión pulsando sobre el icono correspondiente. Una vez abierto se mostrará una ventana como la siguiente:



Tanto el nombre de usuario como el login sólo lo puede cambiar el administrador de su empresa, si por cualquier motivo necesita cambiarlo póngase en contacto con el encargado de dicha tarea. Desde este menú el usuario puede cambiar tanto su contraseña, el número pin de su dispositivo, activar/desactivar la opción de no molestar (para que no suene el teléfono), activar/desactivar el envío de alertas por email tanto para buzones de voz como llamadas perdidas, desviar llamadas al buzón de voz como a un número y configurar el comportamiento de la llamada en espera.

Una vez realizados los cambios deseados basta con pulsar guardar para que queden registrados los datos. Para cerrar sin guardar pulsar en la cruz de la esquina superior derecha para cerrar la ventana. Las formas de contacto son aquellas maneras en las que un usuario puede recibir una llamada. Este menú sólo podrá

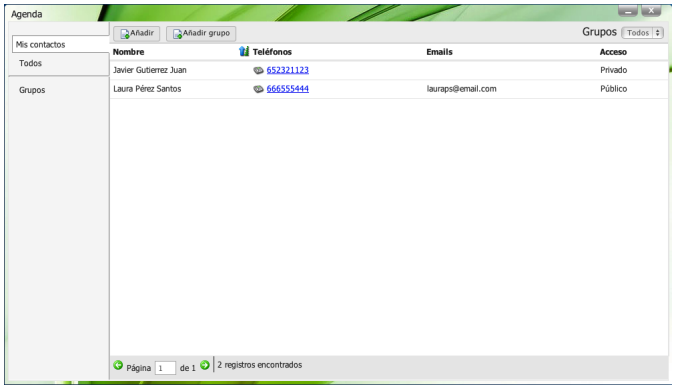
verlo un usuario de perfil avanzado, ya que requiere ciertos conocimientos y no resulta de utilidad para un usuario normal. Una forma de contacto posee:

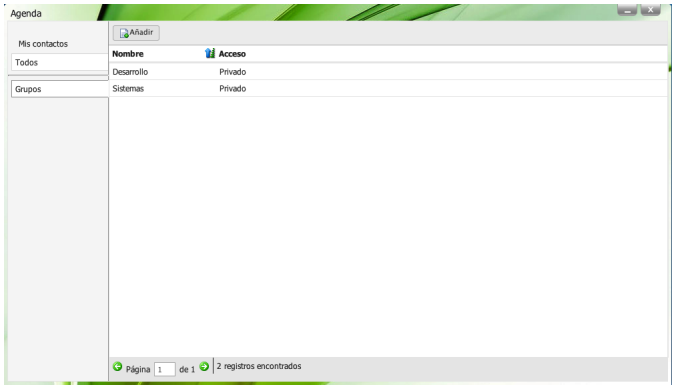
- **Extensión:** Número mediante el cuál el usuario puede acabar recibiendo una llamada.
- **Plan:** El plan aplicado para las llamadas configuradas para el usuario en esa extensión.
- **Tipo:** Una forma de contacto puede ser de tipo ring group, cola o usuario.
- **Ring Group:** La extensión de ring group al que perteneces con la que pueden llamarte.
- **Cola:** La extensión de la cola a la que perteneces y por tanto, sonarán tus dispositivos cuando entre una llamada a dicha cola.
- **Usuario:** Tu extensión (o extensiones) personal(es) que posees mediante las cuales pueden llamarte. Por norma general un usuario tendrá una única extensión personal, aunque hay casos especiales en que esto no es así.
- **Destino:** El destino directo de la extensión. Si es personal será directamente el usuario.
- **Estado:** En caso de que una forma de contacto sea de tipo cola, se mostrará el estado de logueado/deslogueado del usuario en dicha cola.

C.2.2. Acceso a agenda

Para acceder a la agenda pulsamos sobre el icono del escritorio correspondiente. La ventana de la agenda se divide en tres menús: Mis contactos, todos y grupos.

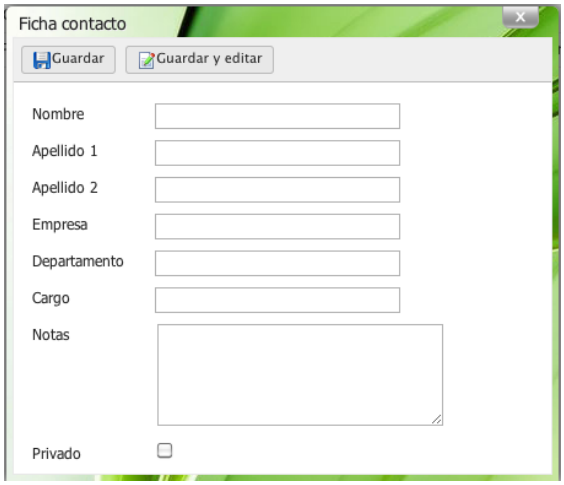
- **Mis contactos:** se encuentran los contactos privados del usuario y que nadie más puede ver.
- **Todos:** se encuentran los contactos públicos para todos los usuarios de la empresa.
- **Grupos:** se encuentran los grupos de contacto. Además tanto en mis contactos como en todos se puede filtrar por grupo para buscar más rápidamente a un contacto que pertenece a un grupo.





Cómo añadir un contacto

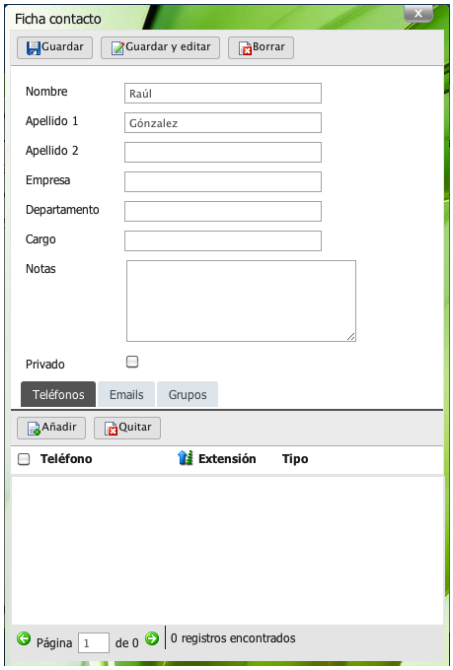
Para añadir un contacto debemos pulsar sobre el botón añadir tanto desde el menú mis contactos como desde el menú todos y se nos abrirá una nueva ventana:



Sólo el campo nombre es obligatorio. Si no lo escribimos no nos dejará continuar y nos saldrá una imagen de aviso dentro del campo nombre indicándonos que el campo es obligatorio.

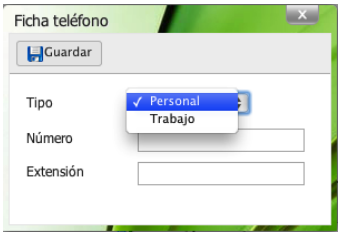
Atención: Si queremos que un contacto sea privado para que sólo pueda verlo el usuario que lo ha creado deberemos pulsar sobre la casilla privado, en otro caso se guardará como público y toda persona perteneciente a la empresa podrá verlo.

Una vez rellenos los campos deseados, podemos seguir editando para introducir teléfonos, emails o asignarle grupos de contacto pulsando sobre "Guardar y Editar". O bien, podemos guardar el contacto sólo con su nombre para editar sus datos en algún otro momento. Si pulsamos sobre "Guardar y editar" se ampliará la ventana con tres pestañas para añadir los teléfonos, emails o grupos a los que pertenece el contacto. Se pueden añadir tantos teléfonos, emails y grupos como queramos.



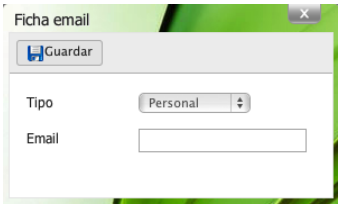
Para quitar un teléfono, un email o un grupo basta con seleccionarlo de la lista correspondiente y pulsar sobre el botón quitar. Se pueden quitar de uno en uno o varios a la vez. Si lo que queremos es eliminar todos deberemos pulsar la casilla de arriba a la izquierda y se seleccionaran todos los teléfonos/emails/grupos dependiendo de la pestaña en la que nos encontremos.

Para añadir un teléfono seleccionamos añadir en la pestaña de "Teléfonos":



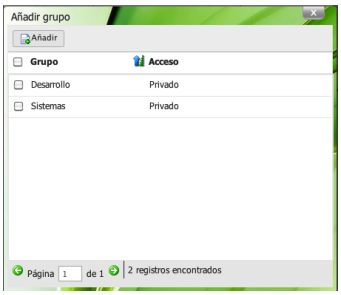
Seleccionamos el tipo de teléfono, por defecto se seleccionará de tipo personal. El número es obligatorio y no nos dejará continuar mientras no se encuentre relleno el campo.

La extensión es opcional, ya que sólo se rellenará en aquella ocasiones en que queramos guardar el teléfono de un contacto de una empresa y sepamos la extensión correspondiente del usuario en esa empresa. Para añadir un email seleccionaremos añadir en la pestaña "Emails":



Seleccionamos el tipo de email, por defecto se seleccionará de tipo personal. El email es obligatorio y no nos dejará continuar mientras no esté relleno el campo.

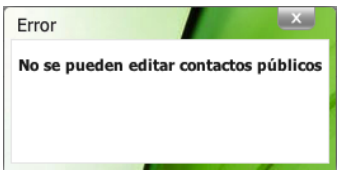
Para añadir un grupo de contacto al que pertenece el contacto, seleccionaremos añadir en la pestaña "Añadir grupo":



Seleccionamos el/los grupo/s de contactos que deseemos añadir y pulsamos en añadir. Si queremos seleccionar todos los grupos pulsaremos sobre la casilla superior izquierda de la tabla y se seleccionarán todas las filas.

Cómo editar un contacto

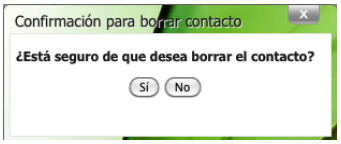
Un contacto sólo se podrá editar si es de carácter privado, o en caso de ser público, solamente si lo hemos creado nosotros en otro caso nos saldrá una ventana informándonos de que la operación no se puede llevar a cabo:



En caso de poder editarlo, se abrirá una ventana igual a la vista en el apartado anterior en la que podremos cambiar desde sus datos, como sus teléfonos, emails y grupos de contacto.

Cómo borrar un contacto

Para poder borrar un contacto, debemos seguir los mismo pasos que para editarlo. Una vez abierta la ventana de edición seleccionaremos el botón borrar situado en la barra superior de la ventana y se abrirá una nueva ventana de confirmación de borrado, para que no podamos borrarlo por accidente.

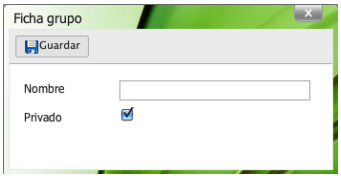


Si no queremos borrarlo, pulsaremos en "No" y la ventana de confirmación se cerrará devolviéndonos a la ventana de edición. Si por el contrario queremos borrarlo, pulsaremos en "Sí" y se cerrará tanto la ventana de confirmación como la de edición y se mostrará la ventana con la lista de contactos actualizada.

Cómo crear un grupo de contactos

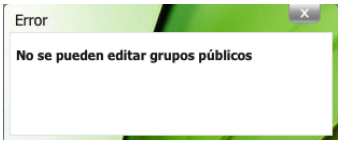
Para crear un grupo de contactos se puede hacer de dos maneras. La primera es desde el menú "Mis contactos" o desde el menú "Todos" pulsando sobre el botón "Añadir grupo". La segunda forma es acudiendo al menú Grupos y pulsando el menú "Añadir". En cualquiera de los casos se abrirá la misma ventana:

Dónde el nombre es obligatorio y por defecto se marcará como tipo privado, si queremos que sea de tipo público deberemos demarcarlo.



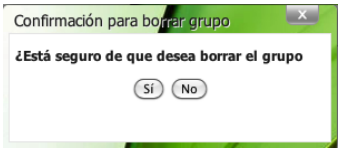
Cómo editar un grupo de contactos

Un grupo sólo se podrá editar si es de carácter privado, o en caso de ser público, solamente si lo hemos creado nosotros en otro caso nos saldrá una ventana informándonos de que la operación no se puede llevar a cabo:



Cómo borrar un grupo de contactos

Para poder borrar un grupo, debemos seguir los mismo pasos que para editarlo. Una vez abierta la ventana de edición seleccionaremos el botón borrar situado en la barra superior de la ventana y se abrirá una nueva ventana de confirmación de borrado, para que no podamos borrarlo por accidente.



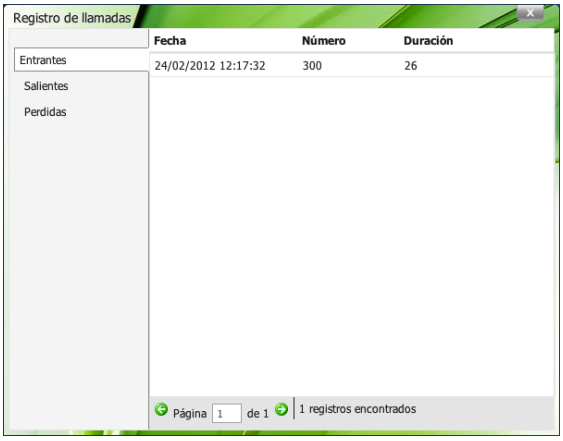
Si no queremos borrarlo, pulsaremos en "No" y la ventana de confirmación se cerrará devolviéndonos a la ventana de edición. Si por el contrario queremos borrarlo, pulsaremos en "Sí" y se cerrará tanto la ventana de confirmación como la de edición y se mostrará la ventana con la lista de grupos actualizada.

C.2.3. Acceso al registro de llamadas

Para acceder al registro de llamadas pulsamos sobre el icono del escritorio de la aplicación. Una vez abierto nos muestra tres menús de llamadas entrantes, salientes y perdidas.

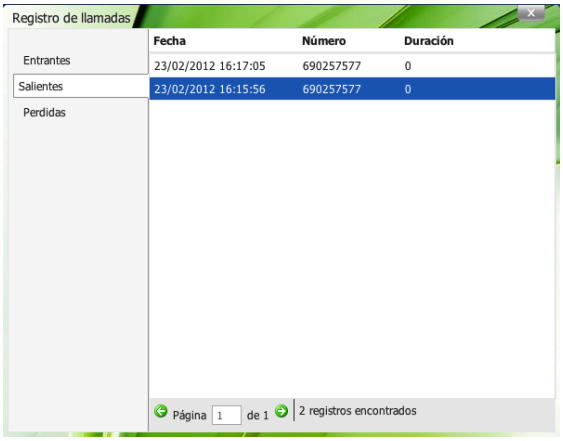
Llamadas entrantes

Por defecto sale abierto en el menú de llamadas entrantes con el resumen de las llamadas entrantes ordenadas por fecha de la más reciente a la más antigua, aunque podemos ordenarla por número o duración (pulsando sobre "número" o sobre "duración" en el titular de la tabla).



Llamadas salientes

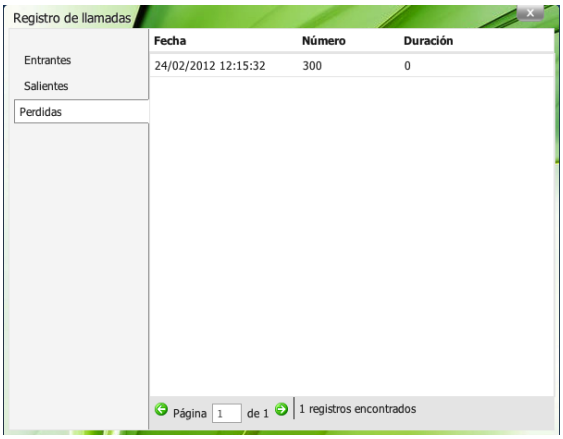
Muestra un listado del registro de llamadas salientes ordenadas de la fecha más reciente a la más antigua, aunque podemos ordenarla por número o duración (pulsando sobre "número" o sobre "duración" en el titular de la tabla).



Llamadas perdidas

Muestra el listado del registro de llamadas perdidas ordenadas de la fecha más reciente a la más antigua, aunque podemos ordenarla por número o duración

(pulsando sobre “número” o sobre “duración” en el titular de la tabla).

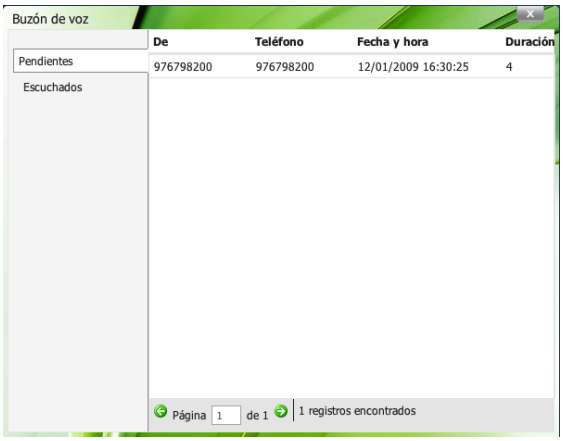


C.2.4. Acceso al buzón de voz

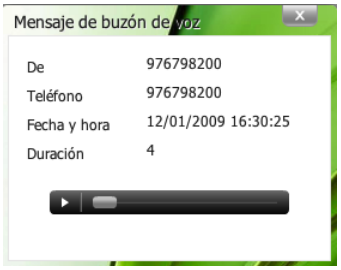
Para acceder al buzón de voz pulsamos sobre el icono del escritorio y se nos abrirá una ventana con dos menús: Pendientes y escuchados. Por defecto se abrirá el de pendientes.

Mensajes de buzón de voz pendientes

El menú de mensajes de buzón de voz pendientes muestra un listado con los mensajes de voz pendientes de escuchar y están ordenador por fecha de la más reciente a la más antigua.

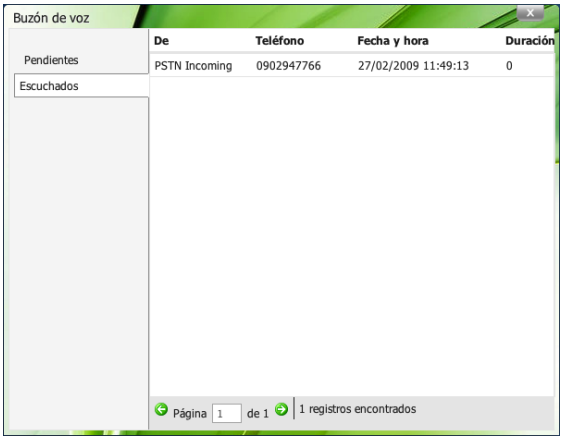


Si queremos escuchar el mensaje deberemos pulsar doble click y sobre el mensaje a escuchar y se abrirá otra ventana con un resumen informativo y un clip de audio para reproducir el mensaje, pausarlo, retroceder, adelantar, etc.

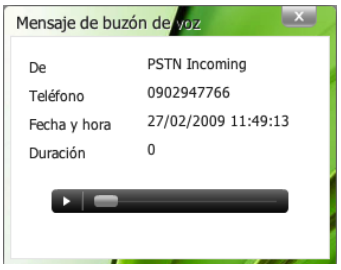


Mensajes de buzón de voz pendientes

El menú de mensajes de buzón de voz escuchados muestra un listado con los mensajes de voz que ya han sido escuchados y están ordenador por fecha de la más reciente a la más antigua.



Si queremos escuchar el mensaje deberemos pulsar doble click y sobre el mensaje a escuchar y se abrirá otra ventana con un resumen informativo y un clip de audio para reproducir el mensaje, pausarlo, retroceder, adelantar, etc.



Apéndice D

Manual técnico

D.1. Aplicación

D.1.1. Introducción

El sistema desarrollado se trata de un frontal Web para configurar, gestionar y operar con un entorno Asterisk. Su principal función es facilitar la tarea de los administradores a la hora de configurar la centralita (PBX) de una empresa, pero a su vez, que sea una herramienta útil para un usuario que quiera consultar su agenda, su registro de llamadas, buzón de voz o cambiar el comportamiento de su dispositivo(por ejemplo, el teléfono de la mesa de su puesto de trabajo).

El sistema a sido desarrollado mediante el uso de un patrón de diseño Modo Vista Controlador (MVC) con el uso de un Framework propio (situado en el directorio /core/es/diaple), lo que nos proporciona una estructura bien definida que nos ayuda a que nuestro proyecto sea organizado y bien desarrollado.

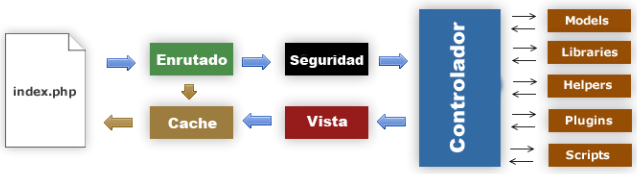


Figura D.1: Arquitectura del Framework propietario

En el archivo de configuración se puede definir dónde se encuentran tanto los controladores como las vistas.

A lo largo de este manual vamos a repasar los principales detalles que forman la aplicación que un técnico debe conocer antes de intentar modificar algún apartado de la aplicación.

D.1.2. El archivo de configuración "app.ini"

En la carpeta raíz se encuentra el fichero de configuración app.ini desde donde se configuran todas las variables de configuración de la aplicación. A continuación detallamos el contenido de este fichero:

[application]

En este apartado se encuentran todas las variables del apartado de configuración de la aplicación.
basePath=/var/www/centralita_v2/public_html/ ;; El path de la base de la dirección url de la aplicación.
appRelativePath=app ;; Path de dónde se encuentra la aplicación.
appRelativeControllerPath=app/controllers/ ;; path dónde se encuentra los controladores
appRelativeViewsPath=app/views/ ;; Path dónde se encuentran las vistas
urlBase=http://desarrollo ;; la url base
urlBasePath=/centralita_v2/public_html/ ;; la url base del path
appManager=index.php ;; Archivo que gestiona y con el que comienza la aplicación.

[init]

En que vista debe comenzar la aplicación.
defaultLanguage=es-es ;; fichero del lenguaje por defecto de la aplicación.
inlineTranslation=false ;; Activar o desactivar la traducción online de la aplicación.

[mysql]

Aquí se encuentran las variables que hacen referencia tanto al servidor donde se encuentran las tablas mysql, con el u
server=desarrollo ;; Nombre del servidor.
user=developer ;; Nombre de usuario de conexión al servidor.
password=developer ;; Contraseña de usuario de conexión al servidor.
database=asterisk_ext_v2.0 ;; Base de datos de la aplicación.
clientCharset=utf8 ;; Codificación de la BD.

[asterisk]

Aquí se encuentran las variables necesarias para conectarse a Asterisk .
→ Está explicado en el apartado de sincronización de la base de datos.

D.1.3. Patrón MVC

Puesto que está basado en un patrón de diseño MVC, la organización de las carpetas tiene por un lado una carpeta llamada controller que es la que contiene los archivos que se encargan de gestionar la parte del controlador y por otra parte la carpeta views que se divide en tantas subcarpetas como vistas existen (una por módulo). Tanto el nombre de la vista como del controlador comienzan en mayúscula seguido de minúsculas. El controlador correspondiente a una vista, tendrá el nombre de la vista seguido de la palabra "Controller" sin espacios. Por ejemplo, la vista "Nombre_de_la_vista" de la carpeta "views" le corresponde el archivo "Nombre_de_la_vistaController.php" de la carpeta "controllers". Cada archivo de un controlador es una clase php que extiende de la clase Controller del FrameWork del MVC. En cada uno de estos archivos se encuentran las funciones que calculan los datos y/o hacen accesos a bases de datos, para posteriormente devolverlos en la vista. Resumiendo, tal y como está desarrollada la aplicación por cada módulo encontraremos al menos un Controlador con su respectiva vista

(carpeta y archivos de las vistas) y un JavaScript con el nombre del módulo que pertenece (en la carpeta "content/js/modules"). Por ejemplo, para el módulo agenda, encontramos su ContactListController (en controllers), una vista ConctacList (subcarpeta en views con sus archivos correspondientes a la vista de la agenda) y un javaScript llamado contactList.js (en content/js/modules) a la que se llamará con base "$.fn.contactList.". Véase el caso de mostrar la ventana del formulario de la agenda tendremos que llamar a la función: "$.fn.contactList.loadDialog".

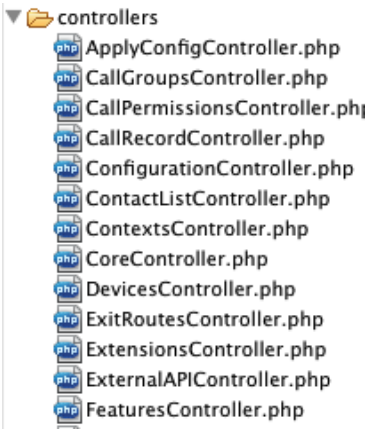


Figura D.2: Carpeta Controllers

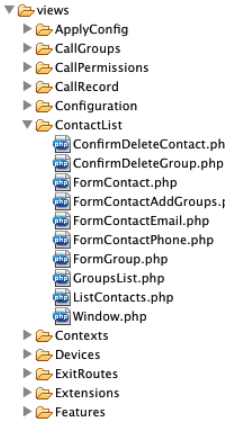


Figura D.3: Carpeta Views

D.1.4. Estructura de la aplicación

La estructura general de la aplicación queda de la siguiente manera:

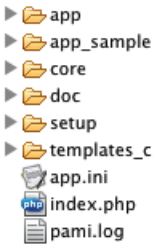


Figura D.4: Estructura general de la aplicación

Donde podemos observar los siguientes archivos y carpetas:

- **app:** se encuentran las carpetas con todo el contenido de la aplicación.
- **app_sample:** Es una aplicación de ejemplo muy sencilla para poder observar como se usan las herramientas creadas (frameworks, helpers, etc) y la estructura del MVC.

- **core:** se entuentra el contenido necesario para el FrameWork de MVC y formularios, así como los helpers para la base de datos, urls y otras utilidades.
- **doc:** se encuentra toda la documentación acerca del framework y los helpers para poder consultarlos comodamente desde la web. Esta documentación ha sido generada con phpDocumentor v1.4.3.
- **setup:** se encuentra un fichero php que se encarga de actualizar tanto la versión de la aplicación como del core que se hayan cambiado remotamente.
- **app.ini:** se encuentran las variables de configuración de la aplicación.
- **index.php:** es la página incial de la aplicación, desde dónde comienza.
- **pami.log:** es el log de sistema de las acciones de la herramienta PAMI.

Si nos centramos en la carpeta app:

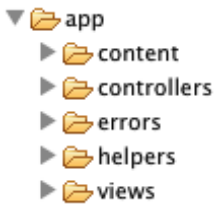


Figura D.5: Carpeta app

La estructura principal de la aplicación (carpeta app) se encuentran las carpetas content, controllers, errors, helpers y views. A continuación explicamos que contenido y función tienen cada una de ellas:

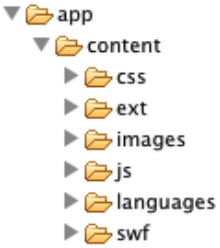


Figura D.6: Carpeta content

En la carpeta content contiene a su vez las subcarpetas:

- **css:** que contiene los estilos de la aplicación Web.

- **Ext:** contenido de aplicaciones externas (véase apartado de sincronización).
- **Images:** contiene las imágenes de la aplicación.
- **Js:** contiene todos los javaScript necesarios de la aplicación.
- **Languages:** contiene el fichero de configuración de los idiomas (por ejemplo, es-es.ini).
- **Swf:** contiene el archivo que carga los elementos flash de la aplicación.

En la carpeta errors se encuentra el archivo php (Exception.php) que se encarga de mostrar las excepciones que se han capturado en la aplicación.

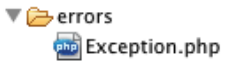


Figura D.7: Carpeta exception

En la carpeta helpers se encuentran los helpers utilizados en la aplicación (explicados en el apartado de sincronización).

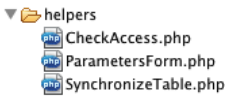


Figura D.8: Carpeta helpers

La carpeta controllers y views está explicada en el apartado de MVC.

D.1.5. Parametrización de formularios en la base de datos

Ciertos formularios de la aplicación se generan consultando a la base de datos. Es el caso de los siguientes formularios:

- Parámetros Cuentas SIP
- Parámetros Trunks
- Parámetros Acciones Planes
- Parámetros Acciones IVRs

Para facilitar la generación de estos formularios y evitar duplicar código se usa el helper ParametersForm. Este helper tiene métodos para obtener los parámetros, generar el formulario y guardarlo.

Para cada formulario en la base de datos tiene que haber una tabla que guarde los parámetros del objeto y otras dos tablas que digan que parámetros y de que tipo debe tener un formulario.

Las tablas de tipos de parámetros y opciones de parámetros son tablas estáticas, es decir únicamente se modifican si queremos cambiar modificaciones los parámetros de un formulario, mientras que la tabla parámetros es dónde se guardan los parámetros del objeto.

En la tabla de tipos de parámetros se guardan los nombres de los parámetros, su tipo, su valor por defecto y su orden. Como veremos más adelante, para ciertos tipos de parámetros son necesarios más datos. Esos datos irán a la tabla opciones de parámetros.

Normalmente nos interesará filtrar en la tabla de parámetros por cierta condición. Por ejemplo en los parámetros de los trunks se filtra por el tipo de tecnología. Para ello el helper permite especificar una condición al obtener los parámetros, más adelante veremos como especificarla.

El formato de las tablas es el siguiente (campos obligatorios):

- Tabla parámetros:
  - parameter_id: referencia a la pk de la tabla tipos de parámetros
  - text_value: para guardar el valor del parámetro
  - number_value: para guardar el valor del parámetro
  - date_value: para guardar el valor del parámetro
  - flag_value: para guardar el valor del parámetro
- Tabla tipos de parámetros
  - name: nombre del parámetro
  - description: descripción del parámetro
  - order_: orden del parámetro
  - required: "S" si es un parámetro obligatorio, cualquier otra cosa si no
  - data_type: Los valores posibles son: text, password, enum, bool y autocomplete.
  - default_value: Valor por defecto del parámetro al generar el formulario si no existe en la tabla de parámetros.
- Tabla opciones de parámetros

- parmeter_type_id: referencia a la tabla de tipos de parámetros
- key: nombre de la opción del parámetro
- value: valor de la opción del parámetro

A continuación se muestran los tipos de datos posibles, qué muestran en el formulario y qué opciones requieren en la tabla de opciones

- text: muestra un input de texto
- password: muestra un input de password
- bool: muestra un checkbox y guarda un "1" si está marcado
- enum: muestra un dropdown con los elementos que digamos en las opciones.
  - Opciones: key = "enum", value = "valor". Tantas veces como elementos del menú.
- autocomplete: genera un campo autocompletable que carga los datos por AJAX de la URL que le digamos.
  - Opciones: dos opciones, controller y action, para poder generar la url.

Una vez las tablas cumplen estos requisitos podemos llamar a los métodos del helper para generar el formulario. Este es un ejemplo de invocación:

```
// Obtener los parámetros
// La condición de filtrado sobre la tabla de parámetros es el último argumento.
$parameters = ParametersForm::getParameters("trunks", $trunk_id, "trunks_parameters", "technologies_types_parameters",
"technologies_types_parameters_options", "technology_type_id = $technology_type_id and trunk = 1");
// Generar el formulario
$form = ParametersForm::create($form,"trunk_id",$trunk_id,$parameters);
// Guardar el formulario
ParametersForm::save("trunks",$trunk_id,"trunks_parameters","technologies_types_parameters",$_POST);
```

D.2. Base de datos

D.2.1. Introducción

En este apartado no nos vamos a introducir en el contenido de la base de datos, si no tan sólo explicar la estructura y cómo funciona la sincronización entre la base de datos de la aplicación y Asterisk.

El sistema está basado en tres bases de datos:

- **Base de datos de la aplicación:** Es la que utiliza la aplicación.

- **Base de datos real time:** Es la que utiliza la centralita. Todos los cambios que se realizan en la aplicación no surgirán efecto en la centralita hasta que no se haga la sincronización.

- **Base de datos de Asterisk:** Es la base de datos de Asterisk.

De esta manera tenemos una sistema de bases de datos completamente independiente y podremos actualizar la base de datos de Asterisk sin tener que modificar la aplicación.

D.2.2. Sincronización de la base de datos de aplicación con Asterisk

El sistema cuenta con varias bases de datos para evitar que la configuración de la centralita que se realiza a través de la interfaz web pase directamente a la configuración de Asterisk. Todos los cambios son aplicados a través del menú especifico para aplicar la configuración. La sincronización de los cambios se realiza a nivel de empresa.

Asterisk tiene su propia base de datos en la que se cargan datos generados a partir de la base de datos de la aplicación. Además de su propia base de datos Asterisk necesita consultar la base de datos de aplicación para ciertas funcionalidades. Por tanto es necesario mantener dos copias de la base de datos de aplicación, una será la que usa la interfaz web y otra, que llamaremos real time, que será la que usa Asterisk.

En el proceso de sincronización los datos pasan de la base de datos de aplicación a la base de datos de aplicación real time y a partir de esta última se generan la información para la base de datos de Asterisk. Toda esta sincronización es modular y siempre filtrando datos a nivel de empresa. Más a delante se detallará el esquema usado para modularizar y gestionar dependencias y se detallarán todos los módulos y qué sincronizan.

En resumen, teniendo en cuenta lo anterior el sistema se compone de tres bases de datos:

- Aplicación.

- Aplicación real time.

- Asterisk.

Y el flujo de datos en el proceso de sincronización será el siguiente:

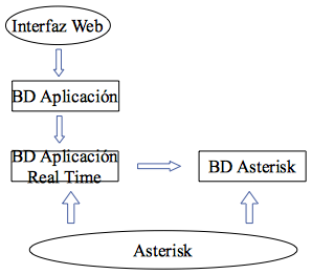


Figura D.9: Flujo de datos sincronización

Configuración

La configuración de la sincronización se realiza en el fichero "app.ini" de la aplicación. En la sección Asterisk de dicho fichero deberemos poner los siguientes datos:

[asterisk]

asteriskip=localhost ; IP o nombre para conectar al Asterisk
asteriskport=5038 ; Puerto de conexión
asteriskuser=admin ; Usuario Asterisk
asteriskpassword=prueba ; Password Asterisk
asteriskdb=asterisk10 ; Nombre de la base de datos de Asterisk
databasert=asterisk_ext_v2.0_rt ; Nombre de la base de datos real time

Subdivisión en módulos

Para permitir mayor control sobre qué cambios se desean aplicar a la configuración real del sistema se puede subdividir en módulos la configuración. Cada módulo cuenta con una serie de parámetros que determinan qué sincroniza y las acciones a ejecutar tras la sincronización. Además un módulo puede especificar si depende de qué otros módulos estén sincronizados.

Entrando en detalle, cada módulo tendrá los siguientes parámetros:

- **Tablas de la aplicación:** Cuando el módulo se sincroniza las tablas especificadas se sincronizan desde la base de datos de aplicación a la base de datos real time.

- **Tablas de Asterisk:** Queries SQL para generar las tablas de Asterisk consultando a la base de datos de aplicación. Además se pueden especificar las acciones a ejecutar tras la modificación de una tabla. La acción como recibe como parámetros los cambios realizados en la tabla, de manera que es posible ejecutar acciones a nivel de registro modificado, permitiendo así diferenciar

inserciones, modificaciones y borrados. La motivación de la acciones a nivel de registro modificado es principalmente la necesidad de ejecutar ciertas acciones en el AMI para algunas tablas.

- **Acción post-sincronización:** Acción ejecutada tras sincronizar las bases de datos. Por ejemplo, generar ficheros, ejecutar comandos en el AMI de Asterisk, etc.
- **Dependencias:** Módulos de los que depende. Una sincronización de un módulo implicará la sincronización de las dependencias.

Gestión de la sincronización

Para gestionar la sincronización desde la interfaz web se proporciona la clase AsteriskConfiguration que ofrece funciones para la consulta de los módulos disponibles. Además también tiene funciones para ejecutar acciones en el AMI. Los métodos públicos disponibles son los siguientes:

- getInstance() → Devuelve la instancia del singleton
- addModule($module) → Añadir un módulo a la configuración. Es un objeto AsteriskModule
- getModules() → Devuelve los módulos que gestiona
- findModuleByName($name) → Devuelve el objeto del módulo por su nombre.
- SendAMIAction($action) → Ejecuta una acción en el AMI de Asterisk

Cada módulo es una instancia de la clase AsteriskModule que implementa una serie de métodos públicos para ser usados desde la interfaz web. Los métodos son los siguientes:

- synchronize($user_id) → Sincroniza el módulo. Recibe un parámetro opcional para especificar si se debe filtrar por los datos modificados por cierto usuario. Si tiene dependencias las sincronizará también.
- onSync() → Devuelve true si el módulo está sincronizado, false en caso contrario.
- getDependecyModules() → Devuelve array con los nombres de los módulos dependientes.
- getAllDependencyModules() → Devuelve un array con los nombres de los módulos recursivamente dependientes (es decir, dependencias de dependencias).

Implementación

A continuación se va explicar cuál es la implementación para la gestión de la configuración en módulos que se ha explicado anteriormente. Básicamente se trata de una clase abstracta (AsteriskModule) que implementa todos los métodos necesarios para sincronizar tablas y gestionar dependencias. Cada módulo será una clase que extenderá de la anterior para realizar la parametrización.

Definición de un módulo (AsteriskModule) Como hemos dicho, implementar un módulo consiste en extender de la clase AsteriskModule e implementar los métodos abstractos, que serán los siguientes:

- getName() → Devuelve el nombre del módulo
- getDependencyModulesName() → Devuelve una array con los nombres de los módulos de los que depende.
- getAppTables() → Devuelve un array con las tablas de la base de datos de aplicación que sincroniza.
- getAsteriskTables() → Devuelve un array con objetos que describen las tablas de Asterisk a sincronizar. Cada objeto tendrá las siguientes propiedades:
  - name: Nombre de la tabla
  - pk: Primary key de la tabla
  - columns: Array con los nombres de las columnas
  - query: Query SQL que genera los datos de la tabla de Asterisk consultando a la base de datos de la aplicación
- postAsteriskTablesSyncAction($table,$changes) → Acción que se ejecuta tras la sincronización de la base de datos de Asterisk. Recibe el nombre de la tabla y un objeto con los registros modificados. El objeto de registros modificados tiene tres propiedades: inserts, deletes y updates. Cada una de las propiedades es un array de objetos que representa una fila de la tabla.
- postSyncAction() → Acción ejecutada tras la sincronización del módulo.

Para dejar más claro todo lo anterior, este es un ejemplo que sincroniza la configuración de cuentas SIP:

```
class SipAccountsAsterisk extends AsteriskModule {

  public function getName () {
    return “Cuentas SIP ”;
  }

  public function getDependencyModulesNames () {
    return array(“Usuarios”,“Dispositivos”,“Contextos”);
  }

  public function getAppTables () {
    return array(“accounts”,“accounts_parameters”);
  }

  protected function getAsteriskTables () {
    // devuele array con las tablas la db de asterisk a sincronizar

    $company_id = Application::getInstance()->getFromSession(“company_id”);
    $tables = array();
    // Tabla sip_users
    $table = new StdClass();
    $table->name = "sip_users";
    $table->pk = ïd";
    $table->columns = array(ïd","name","host","nat","type",“callerid”,“context”, “qualify”,“secret”,
    "disallow",“cc_agent_policy”,“cc_monitor_policy”,“call-limit”,“allow”,“permit”);
    $table->query =
    ’select a.account_id as id, concat_ws("_",lpad(u.company_id,3,“0”),
    lpad(a.account_id,5,"0")) as name, "dynamic.ªs host, "no.ªs nat, "friend.ªs type,
    u.name as callerid,
    (select concat_ws(“_”, lpad(company_id,3,“0”), str2asterisk(name)) from contexts where
    internal = "1.ªnd company_id = ’ . $company_id . ’) as context,zes.ªs qualify,
    u.pin as secret, .ªll.ªs disallow,"generic.ªs cc_agent_policy,
    "generic.ªs cc_monitor_policy, "1.ªs çall-limit",
    – parametros
    group_concat(if(ttp.parameter = çodecs", ap.text_value, NULL)) as allow,
    group_concat(if(ttp.parameter = .ªcl", ap.text_value, NULL)) as permit
    from accounts a join users u on a.user_id = u.user_id and u.deleted_date is null
    left join accounts_parameters ap on ap.account_id = a.account_id and ap.deleted_date is null
    left join technologies_types_parameters ttp
    on ttp.technology_type_parameter_id = ap.technology_type_parameter_id
    where a.deleted_date is null and a.technology_type_id = 1 and u.company_id = ’.$company_id.’
    group by a.account_id’;
    $tables[] = $table;
    return $tables;
  }

  protected function postAsteriskTablesSyncAction ($table,$changes) {
    // acciones ejecutadas tras la sincronizacion de una tabla de asterisk
    // recibe como parametros la tabla sincronizada y los cambios aplicados
    switch ( $table ) {
      case "sip_users":
      // cuentas nuevas
      foreach ( $changes->inserts as $insert ) {
        $name = $insert->name;
        $this->sendAMICmd("sip show peer $name load");
      }
      // cuentas borradas
      foreach ( $changes->deletes as $delete ) {
        $name = $delete->name;
        $this->sendAMICmd("sip prune realtime peer $name");
      }
      // cuentas actualizadas
      foreach ( $changes->updates as $update ) {
        $name = $update->name;
        $this->sendAMICmd("sip prune realtime peer $name");
        $this->sendAMICmd("sip show peer $name load");
      }
      break;
    }
  }

  protected function postSyncAction () {
    // acciones ejecutadas tras la sincronización del modulo
  }
}
```

Sobre el funcionamiento interno

Como hemos comentado anteriormente la clase abstracta AsteriskModule implementa todo el comportamiento común que tienen los módulos. Por un lado sincroniza tablas de aplicación y Asterisk y por otro gestiona las dependencias entre módulos. El método que realiza la sincronización (sychronize), se encarga de realizar todas estas funciones, que resumiendo son las siguientes:

1. Iniciar transacción en la base de datos
2. Generar los datos de las tablas de Asterisk que genera el modulo y sus dependencias en tablas temporales
3. Sincronizar las tablas de aplicación del módulo y de sus dependencias
4. Volver a generar los datos de las tablas de Asterisk y compararlos con los datos guardados en el punto 2. Sincronizar las diferencias a las tablas de Asterisk y guardar los registros cambiados para ejecutar la acción post-sincronización.
5. Commit de la transacción
6. Con los registros cambiados obtenidos en el punto 4, ejecutar las acciones post-sincronización de las tablas de Asterisk para el módulo y sus dependencias.
7. Ejecutar la acción post-sincronización del módulo y sus dependencias.

Respecto al paso 2 y 4, se podría pensar que no es necesario guardar el estado anterior de las tablas de Asterisk, sino que podríamos consultar directamente las tablas para obtener las diferencias. Esto sería posible si la aplicación no trabajase con datos de diferentes empresas. Si consultamos directamente a las tablas de Asterisk deberíamos filtrar por empresa para obtener las diferencias y esto supone un problema ya que cuando cargamos datos en las tablas de Asterisk perdemos la referencia a la empresa y por tanto la capacidad de filtrar. Además tenemos que tener en cuenta que es posible que varios módulos puedan insertar datos en la misma tabla de Asterisk.

El diagrama siguiente muestra los pasos descritos anteriormente, aunque la base de datos de aplicación real time aparece dos veces, únicamente representa la misma base de datos en momentos de tiempo distintos, no dos bases de datos.

La sincronización de tablas de la aplicación se realiza mediante el helper SynchronizeTable, que realiza una sincronización basándose en los campos de auditoría, fecha de creación, usuario, etc. Para la sincronización de las tablas de Asterisk

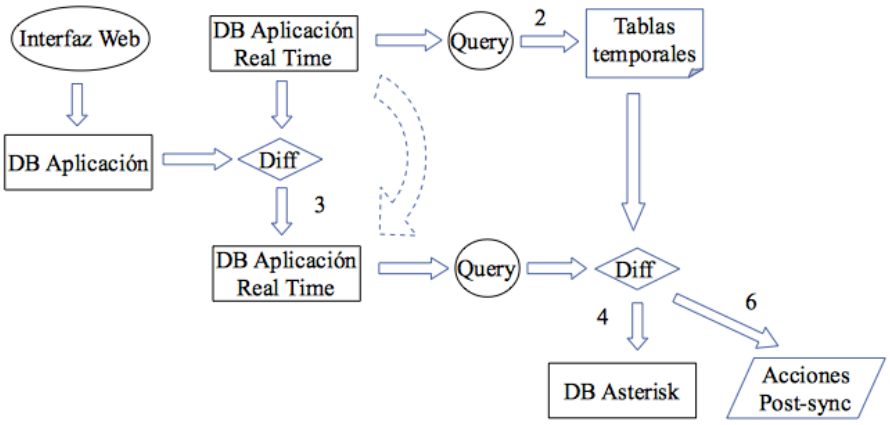


Figura D.10: Sincronización bases de datos

(fase 4) se utilizan los métodos getChanges y syncTable de la clase AsteriskModule. Aquí no es posible utilizar el helper SynchronizeTable ya que no disponemos de campos de auditoría.

Limitaciones

La primera limitación viene dada por el hecho de que MySQL no soporte diferir el chequeo de integridad referencial hasta el commit de la transacción. Por ello, a la hora de especificar las tablas de aplicación que se sincronizan en un módulo es muy importante ponerlas en un orden que permita que la integridad referencial se mantenga en todo momento.

También hay que tener especial cuidado de no establecer dependencias circulares, ya que no están soportadas por la gestión de dependencias y provocarían el mal funcionamiento de la sincronización.

Listado de módulos

En la siguiente tabla se puede ver un listado de los módulos de configuración presentes en sistema con las tablas que sincronizan, las acciones que ejecutan y los ficheros de configuración que generan.

Módulo	Tablas Aplicación	Tablas Asterisk	Acciones y ficheros generados
Permisos de llamada	calls_permissions users_calls_permissions		
Contextos	context		Genera: realtime_context.conf Ejecuta:dialplan reload
Dispositivos	devices_models_manufacturers devices_models devices		
Rutas de salida	exit_routes exit_routes_trunks		
Extensiones	extensions extensions_users extensions_queues extensions_ring_groups extensions_ivrs_groups pickup_groups_extensions extensions_users_plans	extensions	
IVR	schedules ivrs_groups ivrs ivrs_items ivrs_items_parameters ivrs_groups_ivrs ivrs_groups_ivrs_schedules	extensions	Genera: realtime_contexts_ivrs.conf Ejecuta: dialplan reload
Grupos de captura	pickup_groups pickup_groups_users		
Planes	plans actions actions_parameters	extensions	Genera: realtime_contexts_plans.conf Ejecuta: dialplan reload
Colas	queues queues_users_accounts	queues, queues_users_accounts	
Ring Groups	ring_groups ring_groups_users		